Vrije Universiteit Amsterdam

Master Thesis

Horizontal collaboration in transportation and logistics: identifying long-term collaborative partners

---

Author: María Ármann (2776514)

1st supervisor: Rob van der Mei
daily supervisor: Joris Slootweg (CWI)
2nd reader: Vincent Francois-Lavet

A thesis submitted in fulfillment of the requirements for VU Master of Science degree in Business Analytics

August 14, 2024

Abstract

With increasing competition among companies, the transportation sector operates within a highly competitive landscape, emphasising the crucial need for companies to attain maximum efficiency in order to remain viable. One way to do so is for these companies to establish horizontal collaborations. This study aims to develop an algorithm that identifies promising cooperation opportunities for horizontal collaboration. The goal is to create features from logistic data that can capture patterns, enabling the identification of promising cooperation opportunities without needing to solve every possible combination of partners to an exact solution. This is achieved by introducing an integrative method that utilizes all generated features in combination with a machine learning model to predict the expected gains from cooperation. The proposed method proved effective in distinguishing between promising and non-promising collaborations, particularly in ranking collaborations from best to worst. Additionally, the method consistently finds very good solutions for identifying small collaborations involving only two companies.

Acknowledgements

Firstly, I would like to give special thanks to Joris Slootweg for his constant support and excellent guidance throughout the development of this thesis. I would also like to thank Rob van der Mei for his valuable feedback and advice throughout the entire process. Additionally, I would like to thank Vincent Francois-Lavet for serving as my second reader.

Secondly, I would like to extend my sincere gratitude to all my amazing colleagues in the Stochastics group for making my experience at CWI incredibly enjoyable.

Thirdly, I would like to thank NWO, as part of this thesis was sponsored by the Open Technology Program of the Dutch Research Council (NWO), project number 18938.

Lastly, I would like to thank all my close friends and family for their constant mental support throughout the entire process, and special thanks to Annemieke van Goor-Balk for an excellent coordination of the entire internship and for all her invaluable assistance.

Contents

1

Introduction

With increasing competition among companies, the transportation sector operates within a highly competitive landscape, emphasising the crucial need for companies to attain maximum efficiency in order to remain viable. Companies are under constant pressure from the market to increase efficiency in logistics by being able to deliver smaller product quantities more often to meet increasingly shorter customer lead times. Consequently, the average load in logistics typically falls below 100%. The World Economic Forum reports that approximately 27% of goods vehicle kilometers in Europe are traveled without any cargo, and on average, vehicles are typically loaded to only 57% of their maximum capacity [9]. While this is not the optimal utilization of the vehicles, it is also very costly. While companies strive to be competitive, they also seek to minimize costs. This forces companies to reduce the expenses associated with activities that do not add additional value to the operation, such as distributional cost [20]. Additionally, with road freight demand expected to triple by 2050 and increasing pressure for decarbonization to meet the Paris Climate Agreement goals, companies must find ways to boost efficiency and reduce carbon emissions [13]. This forces companies to recognize that possible competitors might be considered as potential collaborative partners. One way to do so is for these companies to establish horizontal collaborations. According to Cruijssen et al. [6] horizontal collaboration is defined as *"an active cooperation between two or more firms that operate on the same level of the supply chain and perform a comparable logistics function on the landside"*. The companies involved can range from being within different sectors of businesses to even being competitors.

Horizontal collaboration offers numerous benefits. Forming these collaborations has the potential to increase load efficiency by merging deliveries from different companies, thereby

reducing the number of partially empty trucks. This improved utilization of trucks means fewer vehicles are needed to handle all deliveries, resulting in significantly lower logistic costs and less congestion on infrastructure. Additionally, collaboration can result in a significant reduction in the total number of kilometers traveled by empty or partially empty trucks. This is mostly obtained by improved utilization of vehicles, but collaboration can also decrease the total traveled kilometers by eliminating unnecessary detours. For instance, if a company's original route included a distant delivery, partnering with another company that has deliveries in that region allows that truck to handle the delivery. This leads to more efficient routes and fewer unnecessary kilometers traveled. Furthermore, collaborative efforts can reduce the number of empty return trips, lowering the percentage of vehicle kilometers traveled without cargo. This is achieved by using these empty return trips to handle deliveries for another company. By increasing the load efficiency and reducing the number of trucks needed, collaboration has the potential to lower overall emissions. Transportation is a significant source of CO_2 emissions, accounting for approximately 2.5% of the EU's total carbon dioxide emissions according to the European Commission in 2019 [11].

According to the *EU transport in figures* report from the European Union in 2021, the total goods transport activities in the EU-27 (the European Union and its 27 Member States) are estimated to amount to 3392 billion tkm, where road transport account for 53% of this total [12]. Therefore, while transportation stands as a primary source of CO_2 emissions, it also serves as the cornerstone of freight logistics, challenging businesses to seek out more effective and sustainable methods for delivering goods.

1.1 Problem statement

Despite the many benefits of horizontal collaboration, such as reduced distribution costs, increased load efficiency, fewer empty return trips, less congestion on infrastructure, and positive environmental impacts, it remains a relatively new research field with limited existing literature. A survey conducted by Dullaert et al., as cited by Cruijssen et al. [7], which reached out to 1500 logistic service providers in Belgium, including 1129 road transportation companies, found that the three major impediments to horizontal collaborations are:

1. Difficult to estimate the savings of the cooperation in advance.

2. Difficulties in finding cooperative partners.

3. Hard to guarantee a fair mechanism for allocating savings for the participants of the collaboration.

Within the limited existing literature on horizontal collaboration, the primary focus appears to be on the third impediment, where researchers often employ cooperative game theory approaches to allocate savings fairly. However, the other two impediments, have received comparatively less attention, where no found articles have been written on finding long-term cooperative partners, creating a gap in research. Therefore, this study aims to find ways to address these impediments, focusing mainly on impediment 2; finding cooperative partners. However, to be able to identify these potential collaborative partners, it is necessary to estimate the benefits of the potential cooperation in advance, as mentioned in point 1. The estimated benefits distinguish between promising and unpromising partnerships. Achieving the optimal collaboration benefits would demand solving the problem precisely for all possible scenarios, covering every possible combination of partners. This implies solving a Vehicle Routing Problem (VRP) for each of these scenarios. Given that VRP problems are NP-hard and thus do not scale efficiently, this approach becomes impractical. Therefore, the main goal of this study is to develop an algorithmic solution capable of efficiently managing large-scale scenarios, including the logistics needs of multiple companies. In this study, the emphasis will be on identifying potential beneficial sub-collaborations. The objective is to pinpoint two or just a few companies that could gain significant benefits from collaborating. This is due to the high costs associated with initiating and maintaining partnerships with other companies and the sensitivity of information sharing, which requires building a substantial level of trust among the participants in the collaboration. With too many companies collaborating, the initial and maintenance cost might eventually outweigh the cost savings obtained from the cooperation. Thus, for practical reasons, the focus will be on identifying small sub-collaborations that have the potential to gain substantially from cooperation.

1.1.1 Capacitated vehicle routing problem for multiple companies

In this study, the primary focus will be on coming up with an approach capable of finding small potential cooperative partnerships. Doing so requires finding near-optimal solutions for the capacitated vehicle routing problem for multiple companies (CVRPMC). The CVRPMC is essentially an extension of the capacitated vehicle routing problem (CVRP),

where vehicles are constrained by limited capacity as they transport items to and from different locations. Each vehicle is tasked with delivering items to specified destinations with pickups limited to a shared depot. These items have capacities, such as weight or volume, and vehicles have a maximum capacity. The objective in this study is to identify potential collaborative partners by addressing the CVRPMC for various combinations of companies, each operating its own fleet of vehicles, with the same capacity.

1.1.2 Assumptions

The purpose of this subsection is to highlight all assumptions that are made in this study. This study aims to find long-term collaborative partners. Similarly to Özener et al. [20] it is assumed that the delivery locations per owner are repeatable each period, as this study focuses on long-term collaborations, meaning that each period each owner will need to deliver goods to the same locations. Additionally, it is assumed that all deliveries are of the same size, specifically one unit each. It is also assumed that all companies involved share the same depot. This assumption covers various cases. For instance, it is common for companies to store their inventories in centralized distribution centers or warehouses, often located in industrial areas. As a result, these warehouses are typically close to each other, making the assumption of a shared depot applicable. Another case involves farmers, whose farms are often located in the same area, meaning their depots are very close to one another. Thus, the assumption of a shared depot is not far from reality in these scenarios. Additionally, as an assumption all vehicles within each instance have the same capacity, meaning that when identifying potential collaborative partners, it is assumed that all companies have access to vehicles of the same size, and that there are always enough available vehicles to handle all the deliveries. This assumption is based on the idea that each company has enough capacity to manage its own deliveries. Therefore, by combining vehicles and deliveries, the collaboration would still have sufficient vehicles to handle all deliveries effectively. Furthermore, in this study, it is assumed that there are no specific time-windows, but that all deliveries for a certain day need to be delivered within that day.

1.1.3 Scope of the study

For the three impediments previously mentioned in section 1.1, the impediment that will be focused on in this study is impediment 2, finding potential partners, where impediment 1, creates the foundation of whether a collaboration is considered promising or not. A known impediments when it comes to collaboration is distributing the gains or costs of the

cooperation, such that no one is better off leaving the cooperation, addressing impediment 3. To achieve this, the evaluation of the identified solution is crucial. This implies that every party involved must be content enough to willingly engage in the collaboration. However, despite impediment 3 being considered one of the three major obstacles to horizontal collaboration, it falls outside the scope of this study. In essence: the scope of this study focuses mainly on finding promising collaboration opportunities, where all participants of the cooperation will gain increased benefits. How to maintain and manage the cooperation, once it has been started, falls out of the scope of this study. Additionally, finding the route, i.e., solving the capacitated vehicle routing problem (CVRP), for the given collaboration of companies will not be the main focus. This implies that optimizing the solution to the CVRP will not be the primary focus, but rather utilizing existing state-of-the-art solvers.

1.2 Aims and objectives

This study aims to create an algorithm that finds promising cooperation opportunities for horizontal collaboration. An opportunity is said to be *promising* if the found solution is preferred by *all* involved parties, such that the total cost with cooperation is at least as bad as without cooperation and preferably better for everyone involved. This includes logistic cost, cooperation cost, as well as lowering traffic congestion by increasing truck load and minimising the number of trucks used.

In essence this study aims to create features from the logistic data that can capture patterns, such that promising cooperation opportunities can be found, without having to solve every possible combination of partners. To this end, the following steps need to be considered:

1. Conduct research on existing literature to determining which methods have been previously used to identify potential collaborative opportunities for horizontal cooperation and determining which of these methods can serve as guidelines or offer insights into possible approaches for this particular study.

2. Use the findings from existing literature to serve as guidelines to discover patterns in the relationships between companies to determine which companies should be considered compatible, such that they have high potential of cost savings.

3. Create features capable of capturing these patterns and utilize them as features in a machine learning model to enable the prediction of potential gains from the possible collaborations.

It should be noted that time restrictions are not necessary when training the machine learning model. However, it is important that new predictions for new instances can be solved within a reasonable time frame to be useful for potential users. In this context, a reasonable time frame could range from minutes to a few days. This is considered acceptable as the potential partnerships are viewed as long-term, making a few days an acceptable duration to identify these partnerships. From the aforementioned objectives, this study aims to achieve the following two outcomes:

1. Being able to correctly predict the highest rank of the potential gains. This means that out of a pool of companies, the algorithm should be able to detect which of these companies would benefit the most from cooperation. It is important that the algorithm's predictions for the highest potential gains accurately reflect the actual highest potential gains. This means that correctly predicting the top gains is more important than accurately predicting the lowest gains, as they will be disregarded.

2. Being able to identify the best combination of collaborations to maximize benefits for all companies. This requires matching each company with a partner in a way that maximizes total gains. In this context, accurately predicting even the lowest gains is important, as all potential collaborations will be considered.

For both outcomes, it is less important that the algorithm manages to predict the exact gains correctly, but more important that the algorithm is able to correctly rank the combinations based on those gains. This means that the highest predicted gains, should ideally be the highest real gains as well as the lowest predicted gains being the lowest real gains. This is because the algorithm should be able to find the most promising collaborations from a pool of companies. It should not miss the most promising collaborations and it should not predict the unpromising ones as promising. However, while slight inaccuracies in predicted gains are acceptable, the predictions should still be representative of the true values. This means negative values should be predicted as negative and positive values as positive. This also implies that if two collaborations have roughly the same benefits, the predictions should reflect this similarity, making the exact ranking within those instances less critical.

1.3 Thesis organization

This thesis report will be structured as follows. Chapter 2 provides the background information for this thesis as well as highlighting key findings from literature review. Chapter

3 offers an overview of the feature generation as well as outlining the modelling approach. Chapter 4 presents all the results obtained from the aforementioned models and finally, Chapter 5 concludes the study and offers insights into potential future research directions.

2

Related research

This chapter aims to provide deeper context of the study as well as providing a review of related research. The purpose of this chapter it to address the first goal mentioned in chapter 1.2, i.e. to determine which methods are most suitable for collaboration forming in horizontal collaboration.

The sections will be divided as follows: firstly, section 2.1 highlights methods from existing literature on finding potential partnerships for horizontal collaboration. Since literature on horizontal collaboration is limited, these methods will primarily serve as inspiration or guidelines. This is because these methods often address problems within horizontal collaboration, although the specific problem in this study may differ slightly. Examples of approaches that slightly differ from this study include auction-based approaches or joint route planning methods. Although the field itself is underdeveloped, valuable insights can be learned from related existing literature. Secondly, section 2.2 summarizes the key findings from literature review that will serve as guidelines in the feature development of this study.

2.1 Finding potential partnerships

Cruijssen et al. [5] conducted a large-scale survey discussing the major opportunities and impediments of horizontal collaboration. The main findings indicated that logistics service providers generally have a strong belief in the potential benefits of enhanced profitability and improved service quality in horizontal collaboration. However, they concluded that the largest impediments include finding a reliable party to lead the cooperation and to construct a fair allocation mechanism for the benefits.

Addressing question 1 from section 1.2 a literature survey will be conducted, determining which methods would be suitable to find potential collaborative partners in horizontal collaboration. In this study, the aim is to determine compatible partners who would make effective collaborators. Once these partners have been identified, the problem will be approached as a joint route planning problem. This means that all orders from participants in the collaboration are combined and efficient route schemes are set up to find the optimal route for all the joint deliveries, using appropriate vehicle routing algorithms. For this study an existing state of the art solver will be utilized for solving the vehicle routing problems. The choice of a solver is further discussed in section 3.1.

In a study conducted by Verdonck et al. [23] they provide a representative and structured review of the scattered amount of scientific research on horizontal logistics cooperation where in that literature they discuss the two main ways in which carriers cooperate horizontally: order sharing and capacity sharing. Most articles on horizontal collaboration are devoted to order sharing, where customer requests or orders are exchanged between the participating organisations through various techniques. However, some of these articles revolve around capacity sharing, where instead of sharing customer requests, carriers cooperate horizontally through the sharing of vehicle capacities.

2.1.1 Order sharing

As stated by Verdonck et al. [23] the majority of articles about horizontal cooperation revolve around order sharing, in particular: auction-based mechanism or joint route planning. However, there are a few articles focusing on finding collaborative partners based on geographical compatibility. The methods focusing on geographical compatibility are more appropriate for this study and will therefore be discussed in more detail.

2.1.1.1 Auction-based mechanisms

In existing literature on order sharing, or horizontal collaboration, some have proposed auction-based mechanisms to form collaborations and, hence, to gain additional benefits for all companies involved. An auction-based mechanism refers to the idea of each individual company first construction the routes for their own deliveries and then allowing other companies to buy space in their carrier.

Song et al. [22] proposed an auction-based mechanism for small and medium-sized delivery companies. In their approach, when a new customer order request comes in, the

delivery company evaluates if it is worth handling the order themselves. If not, they decide on the highest price they are willing to pay another company to do the delivery, referred to as the reservation price. Other delivery companies are notified that the customer order is available for bidding. The other companies then evaluate based on similar methods if they can do the delivery efficiently and at what cost. The original company compares the bids they receive to their reservation price and choose the bid that is the lowest and most acceptable to them.

Similarly, Krajewska et al. [16] suggested an auction-based mechanism that works in the following way: each company gets orders from their customers that they need to deliver. For each order they decide whether it is cheaper to deliver it by themselves or forward it to a subcontractor. They try to pick the option that minimises the total cost. This means that each company figures out the lowest cost for handling each order. This cost is the lower cost out of doing the job using own resources or hiring someone else to deliver it. Then, in the profit optimization phase, customer orders are swapped between cooperating delivery companies to maximize the overall profit of the collaboration. All orders are then grouped together in bundles and the associated cost of handling these bundles is found. They then use a special type of auction to decide which bundle combination will make the most profit. Lastly, these bundles are assigned to the companies involved.

While many articles in the limited literature on horizontal collaboration have examined auction-based mechanisms, these approaches are not well-suited for this study. The focus here is on identifying long-term potential collaborative partners to minimize overall delivery costs for all involved companies. In an auction-based approach, every company is considered a potential partner for each delivery. However, due to the high costs of forming partnerships and concerns about sharing sensitive information, this may not be easily accepted by companies in practice. Therefore, this study aims to identify a small number of partners who can establish long-term collaborations rather than bidding on individual orders.

2.1.1.2 Joint route planning

One way to optimally share orders from collaborative companies, is joint route planning. This method solves a routing problem for all orders from all companies combined, using an appropriate vehicle routing techniques. A literature review conducted by Verdonck et al. [23] shows an overview over multiple papers focusing on joint route planning when it comes to horizontal collaboration.

For a heuristic to find the best combination of companies to form partnerships, Cruijssen et al. [4] suggested a routing heuristic based on the modified application of Clarke and Wright's savings heuristic by Liu and Shen [19]. In Cruijssen's study, the objective is to construct routes that all start and end at a origin destination. For all routes trucks of identical size are used, where each delivery is fulfilled by exactly one truck and each delivery is fulfilled within a specified time window. The heuristic starts with an initial solution, where each route is constructed based on the modified application of Clarke and Wright's savings heuristic by Liu and Shen. In the original Clarke and Wright's savings heuristic, when combining two routes, only the start and the end position of each route is considered for insertion. However, in the modified version, instead of only allowing a route to be joined in the start and the end of another route, it can be inserted at any position of the other route. Figure 2.1 shows an example for two companies, A and B, where cases 1 and 2 show the possibilities for combined routes for the original Clarke and Wright's savings heuristic, while in the modified version cases 3 and 4 are also considered.

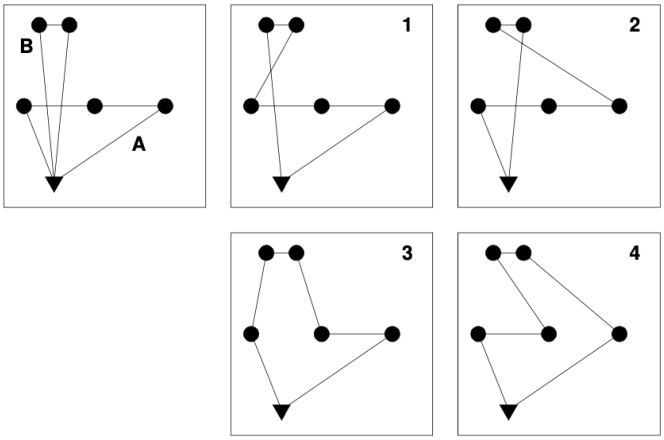

Figure 2.1: The modified application of Clarke and Wright's savings heuristic by Liu and Shen [19].

After the initial routes have been constructed the algorithm then tries to minimise the number of routes by looping through all routes, trying to insert the customers one by one into other routes. A route can be eliminated if all customers of that route can be reinserted into other existing routes. Whether they can be inserted into another route or not is based on a criticality score, where that score is based on demand, time window width and distance from the depot. After the elimination process, two local search operators, ICROSS and IOPT, are executed iteratively until no further improvements of the routes

can be made. These local search operators are similar to the well-known CROSS and Or-opt operators, except that the relocation of segments is also attempted in inverted order. This routing heuristic resulted in 30.7% cost savings on their given benchmark instance and 43.2% improvement in the load of trucks.

According to the same literature review study conducted by Verdonck et al. [23], since joint route planning problems are a NP-hard problem, most current studies on horizontal collaboration in that setting have a heuristic nature. Interestingly, according to Cruijssen et al. [4], they found out that joint route planning is more profitable for small transport companies than for larger ones. They also found out that joint route planning is more profitable in sectors where on average the orders are smaller compared to the truck's capacity and that joint route planning is more profitable where customers are located across a larger region (such as Europe), compared to when they are located quite close to each other (such as in regional distributions). In terms of market share, they also found out that the synergy value decreases when the market share is less equal over the participants in the coalition. This is due to the fact that in a scenario with one leading company, with the majority of the market share, that company will be able to construct very efficient routes, even without joint route planning.

As stated in Verdonck et al. [23] by using joint route planning, scale economies may result in reduced travel distances, less empty vehicle movements and number of required vehicles by merging the distribution regions of all collaborative partners. However, even though a few articles have been written about joint route planning in horizontal collaboration and it has shown very promising results of benefits, the downfall of that one is that it assumes that potential partnerships have already been formed, or allows for all companies to be a part of any route. In this study the focus is on finding long-term partnerships where the potential gain of the partnership is the highest. For that reason joint route planning can not be used to find potential partnerships, but to construct the routes for the companies within a partnership, after the partnership has been formed. It can also be used as a proxy to give ideas of good potential partners, or to serve as a baseline for maximum possible gains.

2.1.1.3 Heuristics based on geographical compatibility

This study focuses on identifying potential long-term collaborative partners. For this reason, neither auction-based mechanisms nor joint route planning approaches are suitable. Auction-based mechanisms do not involve long-term partnerships, while joint route planning considers all companies as potential partners for any route. In this study, long-term

partners are limited to only a few companies to avoid increased complexity and collaboration costs that may outweigh the benefits of collaboration, making joint route planning not an appropriate fit. To identify potential collaborative partners that would yield the highest gains, it is important to assess the compatibility of companies. One way to assess compatibility is by examining the geographical locations of their deliveries.

Geographically compatible shipments Creemers et al. [3] created a heuristic that focuses on identifying potential collaborative shipment opportunities based on their geographical compatibility. The algorithm aims to find bundling, back-hauling, and round-trip opportunities based on identifying shipments with similar origins and destinations. Figures 2.2(a) to (c) show a graphical representation of these collaborative opportunities. Bundling opportunities are found if the origins are within a radius r of one another and all destinations of the deliveries are within a radius r of one another. Similarly, back-hauling opportunities require that the the origin and destination of two shipments lie within a radius r of one another and vice versa. A round-trip opportunity then requires the destination and origin to be within a distance r of the respective origin and destination of other partnering lanes. Essentially, the problem involves identifying shipments with similar pickup and delivery locations.

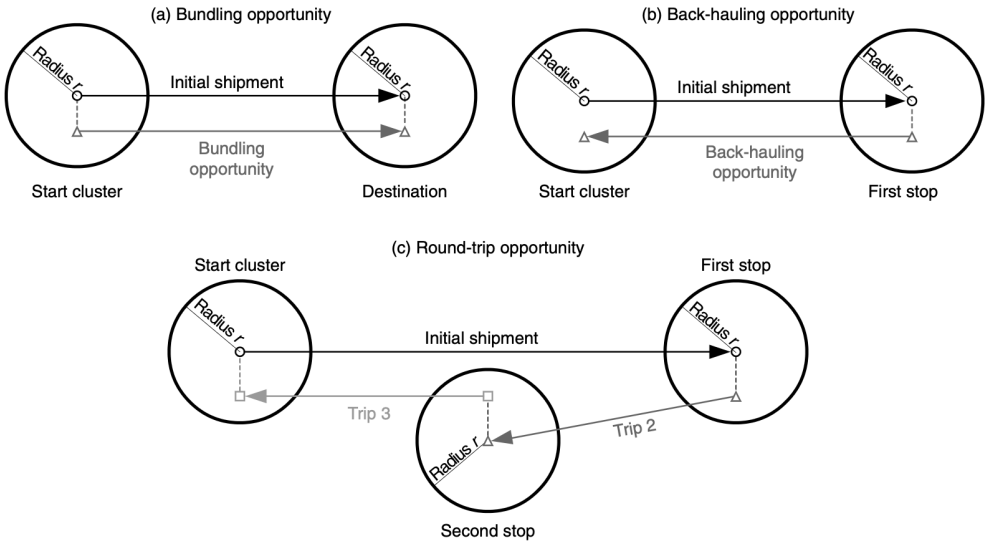

Figure 2.2: Illustration of bundling, back-hauling, and round-trip opportunities where the origin or destination points of different trips are within a small distance (r) of each other. The circles represent the original deliveries, while the triangles and rectangles indicate collaboration opportunities within distance r of each other [3].

This algorithm can serve as an inspiration to this study. However, in their study, they aim to discover compatible deliveries, while in this study the aim is to find compatible companies, where each company has multiple deliveries. Considering the geographical aspect of all deliveries associated with each company could greatly enhance the search for potential partners.

Bounding-box method A part of the algorithm created by Creemers et al. [3] uses a bounding-box method to filter the lanes and to identify potential partners. They created a tool where the goal was on finding geographically compatible shipments that can share the same means of transportation. The bounding-box method, commonly used in computer vision and image processing, involves drawing a rectangular box around objects of interest. In their research, this method was adapted to create such a region and exclude lanes falling outside of it. In their approach, for each delivery, they first created a list of potential partner lanes for bundling opportunities. Since these lanes are only sorted on the latitudinal coordinates of the deliveries, it is likely that some of these candidate lanes are not close to the delivery at all. To filter those lanes out, they created a bounding-box around the delivery and only calculate the distances to those lanes that fall within the box. This approach can sever as an inspiration basis for developing features that employ bounding boxes to identify compatible companies in this study. This means that for each company a bounding box can be created around or between the regions it has deliveries in, allowing the identification of other companies with deliveries that fall within those boxes.

Greedy heuristic Adenso-Díaz et al. [1] also used the idea of geographical compatibility. In their research they aim to find out which deliveries are compatible, both in terms of location and time windows. They operated under the assumptions of enough availability of large trucks and a sufficient number of trucks.

They create their objective function such that they minimise the cost of each delivery, including the discount in total cost that is achieved by adding each new delivery to the same route. The objective function also captures the cost of joining two deliveries. The objective function is defined as:

$$\min \sum_{r} \sum_{k} [1 - \alpha_k(n_r - 1)] \sum_{i} c_{ik} z_{ri} x_{rk} + \sum_{r} \sum_{i} \sum_{j} \sum_{k} y_{rij} z_{rk} v_{ijk}$$

Table 2.3, shows the notation of the model.

Table 1
Notation used in the modelling of the problem.

J	Set of deliveries
i, j	Indices of deliveries
i_r	First (fictional) delivery for route r (no truck, or cost associated).
r	Index for routes
k	Index for vehicle types
W_k	Capacity of vehicles type k
c_{ik}	Transportation cost of delivery i using vehicle type k (including empty return)
Q_i	Load to transport in delivery i
δ_{ij}	$= \begin{cases} 1 & \text{if delivery } j \text{ can be linked after delivery } i; (\delta_{i,j} = 1\ \forall j) \\ 0 & \text{otherwise} \end{cases}$
v_{ijk}	Cost of linking j after i using a vehicle type k (v_{ijk}= when $\delta_{ij} = 0$; for the first fictional delivery of each route, it is $v_{i,j,k} = 0 \quad \forall j \forall k$). It includes distance and time needed for the linking
α_k	% discounted for each additional delivery linked, using vehicle k, as a result of avoiding an empty return
M	Maximum number of linked deliveries that a driver would accept in just one route
x_{rk}	$= \begin{cases} 1 & \text{if route } r \text{ uses a vehicle type } k \\ 0 & \text{otherwise} \end{cases}$
z_{ri}	$= \begin{cases} 1 & \text{if delivery } i \text{ is assigned to route } r \\ 0 & \text{otherwise} \end{cases}$
y_{rij}	$= \begin{cases} 1 & \text{if delivery } j \text{ follows delivery } i \text{ in route } r \\ 0 & \text{otherwise} \end{cases}$
n_r	Number of deliveries forming route r

Figure 2.3: Notation of the model proposed by Adenso-Díaz et al. [1].

The first summation captures the total cost of each delivery while the second summation captures the cost of joining two deliveries. In the constraints they make sure that each route is served by only one vehicle, as well as respecting time constraints. The constraints are defined as follows:

$$\sum_k x_{rk} = 1 \quad \forall r \tag{1}$$

$$y_{rij} \leq \delta_{ij} \quad \forall r, \forall i, \forall j \tag{2}$$

$$\sum_r \sum_{i \cup i_r} y_{rij} = 1 \quad \forall j \neq i_r, \forall r \tag{3}$$

$$z_{rj} = \sum_{i \cup i_r} y_{rij} \quad \forall r, \forall j \neq i_r \tag{4}$$

$$\sum_k W_k x_{rk} \geq Q_i z_{ri} \quad \forall r, \forall i \tag{5}$$

$$n_r = \sum_i z_{ri} \leq M \quad \forall r \tag{6}$$

Constraint 1 makes sure each route is served by exactly one vehicle and constraint 3 assigns each delivery to exactly one route. Constraint 2 makes sure a delivery will only follow another one if is it feasible by the compatibility matrix, δ. Constraint 4 ensures that if a delivery j follows a delivery i on the same route r, then the binary decision variable reflects that delivery j is also on route r. Lastly, constraint 5 ensures the capacity of the

vehicle is sufficient for all the deliveries on the route and constraint 6 bounds the route length. The main idea behind the model is that the longer the routes the higher the discount. The model itself is a quadratic integer programme which they solved using a greedy heuristic, GRASP (Greedy Randomized Adaptive Search Procedure). GRASP is a two-phase metaheuristic including solution generation and solution improvement. The first step includes generating a solution from scratch by greedily joining different deliveries when possible and profitable to build different routes. The second step consists of improving the generated solution using a local search. They found out that the benefits, or gains, of cooperation diminish gradually as the size of the partnership increases. In other words, as more parties are involved in the cooperation, the benefits gained from it become slightly less significant. Additionally they found out that regardless of the number of companies involved, the benefits obtained from cooperation increase as the percentage of savings from route lengthening increases. As the percentage of savings from route lengthening increases, it means that there are more savings achieved by extending or optimizing the routes taken for deliveries. In other words, if you can make deliveries more efficiently by taking longer routes that save time, fuel, or other resources, the percentage of those savings increases. This indicates that the cooperation is becoming more beneficial as more efficient routes are used, leading to greater overall savings. Additionally, these benefits decrease as the cost of linking any two deliveries increases. In essence, it highlights two factors influencing the gains from cooperation: the efficiency gained through route optimization and the cost associated with linking deliveries, both of which impact the overall benefits.

2.1.2 Capacity sharing

In addition to sharing customer orders, companies can also collaborate by sharing vehicle capacities. Horizontal collaboration in this context has received less attention in research compared to order sharing. The cost of investing in vehicles can be high, so sharing vehicle capital investment among multiple companies can help spread the cost and increase overall vehicle utilization.

Liner shipping carriers As mentioned in the literature review conducted by Verdonck et al. [23], only a few articles have been written on the topic of capacity sharing in the context of horizontal collaboration. However, not all of them are in the context of road transportation. Agarwal et al. [2] studied the problem in the context of ships. In their approach all companies pool their fleets to operate them together and to share the capacity on the ships. Each company decides which routes their shipments should take and then

assigns ships from the shared pool to those routes. This way, the space on each ship is divided among the companies. Sharing capacity helps increase utilization of the ships. This way companies can offer more frequent trips to customers.

Truck capacity sharing Other articles focus on capacity sharing within the context of road transportation. Hernandez et al. [15] studied how multiple trucking companies can work together to share their trucks and reduce costs. In their approach they study the problem in a dynamic environment, where the availability of trucks varies over time. In this study, each company first utilizes their own fleet and then, if available and needed, share trucks with others. When they share trucks, they split the costs of loading, unloading, and holding the goods. The problem is formulated as a linear programming problem with the objective of minimising the total cost of all participating companies. The article concludes that there are clear benefits of collaboration, however as the problem is formulated with availability being dynamic over time, each company needs to take into account the trade-off between lower costs and longer waiting time or higher costs and shorter waiting times.

This study approaches the problem from an order sharing perspective. Since these articles focus on capacity sharing, they do not directly apply to this study.

2.1.3 Clustering vehicle routing problem

The algorithm proposed by Creemers et al. [3] discussed in section 2.1.1.3 can be used as an inspiration to find round-trip opportunities, but instead of using it for bundling and back-hauling opportunities, clustering algorithms can be used instead. Since the algorithm aims to find independent deliveries that are geographically compatible, it might be too complex for identifying compatible collaborative partners where all deliveries are compared. In such cases, clustering algorithms could prove useful.

Three-step heuristic approach The Clustered Vehicle Routing Problem (CluVRP) represents a modification of the classical Vehicle Routing Problem (VRP), centered on the concept of clustering deliveries and addressing the VRP for these clusters. Dondo et al. [10] proposed a heuristic method that addresses this problem through a three-step heuristic approach. Initially, the focus lies on identifying suitable clusters of deliveries. Subsequently, the clusters are aggregated into single nodes, and the VRP is solved to determine which clusters should be assigned to the same route. The last step then focuses on determining the optimal route within the clusters. In this study, the first phase of the

three-step heuristic approach is the one of highest interest. This is the clustering part of the algorithm. As Dondo et al. [10] mentioned, finding a good set of clusters can be quite challenging. In their paper, the goal of clustering is to identify a small set of feasible clusters. When assigning orders to a cluster, the following conditions are kept in mind.

- All orders within a cluster need to fit in a vehicle.
- There exists a route within the cluster that connects the nodes in such a way that all time constraints are satisfied.
- The waiting time for a vehicle due to early arrivals at pick up or delivery locations should be minimised.
- The average length traveled within the cluster should remain low.

Before adding a new order to a cluster, it is ensured that these conditions are met. If these conditions are exceeded, the node is not added to the cluster. In the mathematical formulation of this the *cluster service time* and *cluster time window* are introduced. The earliest service time at a cluster C_n is aC_n which is given by $\min_{i \in C_n}(a_i)$ where a_i is the earliest service time at node i. The latest service time for cluster C_n is bC_n, where bC_n is obtained by taking the minimum of the current bC_n (before node i is added) and the latest service time for node i, b_i. Every time a node is added to the cluster the cluster time window is updated. In this way, the time windows for the all nodes within the cluster are satisfied and the cluster time windows become much tighter. However, the cluster service times is a good approximation of the total time is takes to visit all nodes within the cluster, taking into account travel and idle times throughout the cluster. To choose the most cost efficient vehicle, the first available vehicle with the lowest ration $\frac{q_v}{cf_v}$, where cf_v is the fixed cost for using vehicle v and q_v is capacity of vehicle v, is assigned to the cluster. The travel times between clusters are estimated by calculating the distance between the centroids of the clusters. This heuristic approach effectively reduced the problem's complexity, enabling solving a problem of 100-customer VRPTW to optimality, surpassing the limitations of their original approach of solving a mixed integer linear programming (MILP) to optimality, which was constrained to 25 nodes. It is important to mention that time windows are significant in this article, influencing both cluster construction and route planning between clusters. However, in our study, time windows are not considered, which simplifies the approach.

DBSCAN+ Li et al. [17] proposed a similar algorithm. They also proposed a three-step heuristic algorithm to solve vehicle routing problems with workload balance. Their optimization framework consist of these three steps:

1. Clustering: All orders are grouped into clusters utilizing a new clustering scheme, named DBSCAN+. The DBSCAN+ is proposed to detect clusters and noises of arbitrary shape in location data. This is a clustering algorithm combining the improved density-based spatial clustering of applications with noise (DBSCAN) [18] and a Gauss mixture model. First the Gauss mixture model is utilized to fit the probability distribution of the dataset to determine different density levels of the clusters, then based on the DBSCAN the subdatasets are clustered based on different density levels. The final clusters are called microclusters.

2. Microcluster fusion: The microclusters are combined to form clusters that balances the workload of the vehicles.

3. Route search: The final step includes constructing the final routes for the vehicles utilizing a modified Ant Colony Optimization algorithm (ACO), named ACO+.

As the microcluster fusion step is used to take into account the workload balance and make sure that all vehicles have equal amount of workload, this part is irrelevant to this study. Similarly, route search is used to construct the final routes for all vehicles. This step is also unnecessary in this study. The clustering step is of main interest. The authors state that the original DBSCAN clustering algorithm has the benefits of being able to find clusters of arbitrary shape and it can deal with noise in the location data. DBSCAN clustering algorithm has often been used in pattern discovery of large location data. However, the DBSCAN has some shortcomings, where it has low processing speed with large data and it has difficulties when applied in practice where the location data is not uniform. Therefore their study focuses on the improved DBSCAN, the DBSCAN+ algorithm, where the data is first divided into subdatasets, where each subdataset is created based on the Gauss mixture model, before being clustered.

2.1.4 Company characteristics

In horizontal collaboration the extent of the profit obtained by cooperation is highly dependent on the characteristics of the partners forming the collaboration and the characteristics of their operations. Cuervo et al. [8] conducted a simulation study to analyse these effects. They concluded that the characteristics of the partners within the cooperation have

a large impact on the performance of the collaboration as a whole. The study focuses on two-partner collaborations. In the study they both look at which characteristics have the highest impact on the profit, but also which combination of characteristics between companies lead to the most profitable collaborations. The study concluded that the interaction between characteristics and how the characteristics of two partners complete each other have a major influence on the performance of the cooperation. The study led to three main insights that can play useful when finding potential collaborative partners.

- The average order size is the most influential characteristic. Companies with different order sizes benefit the most. One company that carries big orders that, in its own plan, would leave trucks with extra space. Another company that carries small orders that, in the joint plan, can use the extra space in the partner's trucks.
- The number of orders a company needs to transport also significantly affects the profit made by the cooperation. The more orders a company has, the greater its potential to create a more efficient plan when working with a partner.
- The company's ability to delay its orders has a less significant impact on the cooperation's profit compared to the other characteristics. This can play beneficial when a company has many trucks with unused capacity. This flexibility allows the joint plan to utilize that extra capacity better. However, if the company itself can make use of this flexibility to improve its own standalone plan, the benefit of collaboration may decrease significantly.

2.2 Key findings from related research

When it comes to horizontal collaboration, literature is still very scarce. As previously mentioned, most articles addressing the identification of potential collaborative partners focus on order sharing, typically through joint route planning or auction-based mechanisms. However, since the objective of this study is to identify long-term collaborative partners, these approaches are not ideally suited. In the scarce literature on horizontal collaboration, identifying long-term collaborative partners creates a gap in research. Therefore, this study aims to address that gap by using these previously mentioned approaches as inspiration for further work.

3

Methods

According to the objectives outlined in section 1.2 and the findings from the literature review in chapter 2, this chapter aims to provide an overview of the rationale behind the algorithms developed for identifying potential partnerships within horizontal collaboration. As this study focuses on finding the potential partnerships, optimising a vehicle routing solver, falls outside the scope of this study. However, finding a suitable vehicle routing algorithm still falls within the scope. Section 3.1 discusses state of the art solvers for vehicle routing problems that can be utilized in this study. Section 3.2 covers all the feature generation steps taken to create additional features aimed at discovering patterns within the data that can help identify beneficial partnerships. Sections 3.3 and 3.4 introduces the machine learning models that will be used to predict potential gains from these partnerships, as well as providing information on the predicted value. The final model used in this study is a iterative machine learning method. This method utilizes all features created in the feature generation step, as well as the machine learning models. Section 3.5 provides details on the iterative method. Lastly, section 3.6 describes the last step of the algorithm. The last step is to consider assignment problems. This means that when all predictions have been made then we want to assign every company to a cooperation such that the total gain is maximised.

3.1 Vehicle routing algorithms

This study focuses on identifying potential partners who can benefit the most from collaboration, rather than on constructing exact routes in vehicle routing problems. Therefore, an existing algorithm will be utilized. The following sections will discuss the shortcomings

and the benefits of some existing algorithms and why they should, or should not, be chosen as a tool utilized in this study.

3.1.1 OR-Tools

OR-Tools is a Python package used to solve vehicle routing problems (VRPs) with exact methods. VRPs are NP-hard, meaning that solving even small instances can be extremely time-consuming, and larger instances can quickly become infeasible. In this study, OR-Tools was tested to find optimal solutions for smaller instances, but it proved too slow even for the smallest cases. Since this study aims to evaluate how well the algorithm for identifying potential partnerships scales, this approach is not ideal. The chosen vehicle routing solver should efficiently solve instances of various sizes within a reasonable time frame to be able to compare the identified ideal partnerships with the optimal solutions.

3.1.2 PyVRP

Another existing vehicle routing solver is PyVRP, which heuristically solves various versions of the vehicle routing problem (VRP). It supports the capacitated vehicle routing problems (CVRP), making it relevant for this study. PyVRP is a polished implementation of the algorithm that ranked 1st in the 2021 DIMACS VRPTW challenge and, after improvements, ranked 1st on the static variant of the EURO meets NeurIPS 2022 vehicle routing competition. At its core, the implementation consists of a genetic algorithm, a population, and a local search improvement method.

The advantages of using PyVRP include how easy it is to implement and the flexibility it offers for customization to meet the specific requirements of this study. However, PyVRP has some limitations, particularly regarding the consistency of solutions. Specifically, for a given set of deliveries for one company, PyVRP may produce visually suboptimal routes where the path crosses itself. In Euclidean space, a crossing route indicates a less efficient path because the shortest distance between two points is a straight line. A self-crossing route suggests that a simpler, non-crossing path exists that would cover a shorter distance. On the other hand, when the same company collaborates with another and the deliveries are combined, PyVRP is able to find a solution without crossing paths. Figures 3.1 and 3.2 illustrate such cases. Figure 3.1 displays the best route found for company 4, with the stopping criterion set to 100000 iterations if no better solution was found. Generally, in this study, the best solutions were achieved with a stopping criterion of 300 iterations, except in a few cases where improvements stopped improving after 500 iterations. This

means that for those cases where a visually suboptimal route was obtained, even after 100.000 iterations, the found solution remained unchanged, indicating it might be the best achievable solution from the heuristic approach. This could result in the algorithm overestimating the gains from cooperation. Nonetheless, while this limitation may affect the accuracy of exact solutions, it occurs only in a very few instances, making PyVRP still a viable option for approximating the VRP between clusters. The primary advantage of using PyVRP is its efficiency in solving the problem quickly.

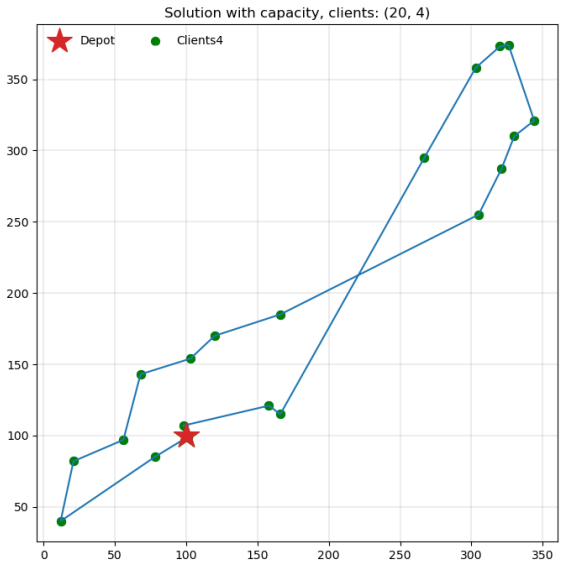

Figure 3.1: Noticeably sub-optimal route for 100.000 iterations.

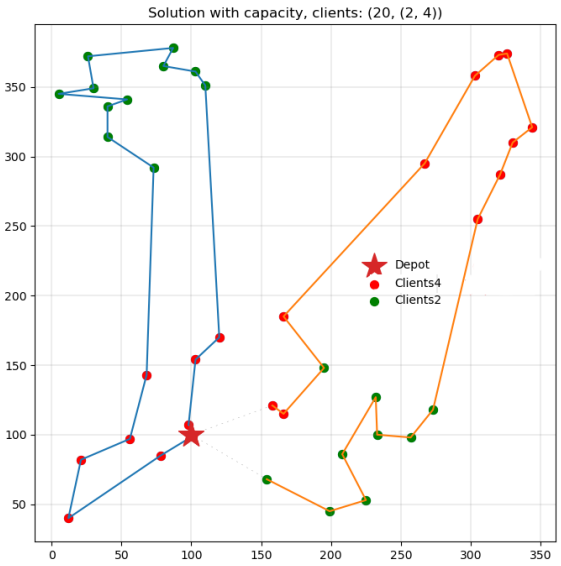

Figure 3.2: Best found route for cooperation

When using PyVRP, a stopping criterion is essential to prevent excessively long solution times. The available options include: *MaxRuntime*, *MaxIterations* and *NoImprovement*. *MaxRuntime* stops the algorithm after a specified time limit (in seconds), *MaxIterations* halts it after reaching a set number of iterations, and *NoImprovement* ends the process if no better solution is found within a fixed number of iterations. A key advantage of PyVRP is the ability to use these stopping criteria in combination. In this study, the *NoImprovement* criterion was applied with a limit of 500 iterations. If this approach led to excessively long runtimes, *MaxRuntime* of one minute, was also employed to ensure the algorithm did not get stuck.

3.1.3 VRPy

A third solver is VRPy. VRPy is a python framework used for solving instances of various types of Vehicle Routing Problems (VRP) including the Capacitated VRP (CVRP), making it equally as suitable for this study as PyVRP. VRPy employs a column generation approach to solve vehicle routing problems. This method involves iteratively generating routes (or columns) through a pricing problem, which are then submitted to a master problem. The master problem selects the optimal routes from the pool to ensure each vertex is serviced exactly once. Results from the master problem guide the search for new routes that may improve the solution's cost, continuing the process iteratively.

Due to the limitations observed with PyVRP, VRPy was tested to determine if it could address those issues. However, like PyVRP, VRPy does not guarantee an optimal solution and, after implementation, exhibited the same limitations as PyVRP.

Similar to PyVRP, VRPy requires a stopping criterion. The stopping criterion for VRPy is solely a time limit. However, unlike PyVRP, which continues to run until the time limit is reached, VRPy will return the solution as soon as the best one is found, avoiding unnecessary computation on smaller instances.

3.1.4 Chosen vehicle routing solver

Both PyVRP and VRPy share similar limitations, occasionally producing visibly suboptimal solutions. For instance, figure 3.3 shows the best route obtained for customer 1 from the instance "homberger_1000_customer_instances/R2_10_1.TXT" involving 10 companies and vehicle capacities of 20, with instances ranging from 400-600. Both algorithms were executed for 2 hours. VRPy completed 541 iterations without finding an improved solution. Similarly, PyVRP ran for over 1.000.000 iterations without any improvement. It is important to note that iterations are defined differently in each algorithm. However, the fact that even a small vehicle routing problem cannot be effectively solved in 2 hours highlights a major issue. Given that each large instance consists of up to 2000 smaller cases and there are up to 200 large instances overall, the total computation time would exceed 80.000 hours, or over 9 years, to find a satisfactory solution. Thus, it is essential to balance runtime with the quality of the solutions found.

Fortunately, this study benefits from access to a supercomputer, which enables running more instances in less time and allows for simultaneous execution of multiple instances. With this capability, if each small vehicle routing case can be solved within 10 seconds, larger instances with up to 2000 cases could potentially be solved within 6 hours.

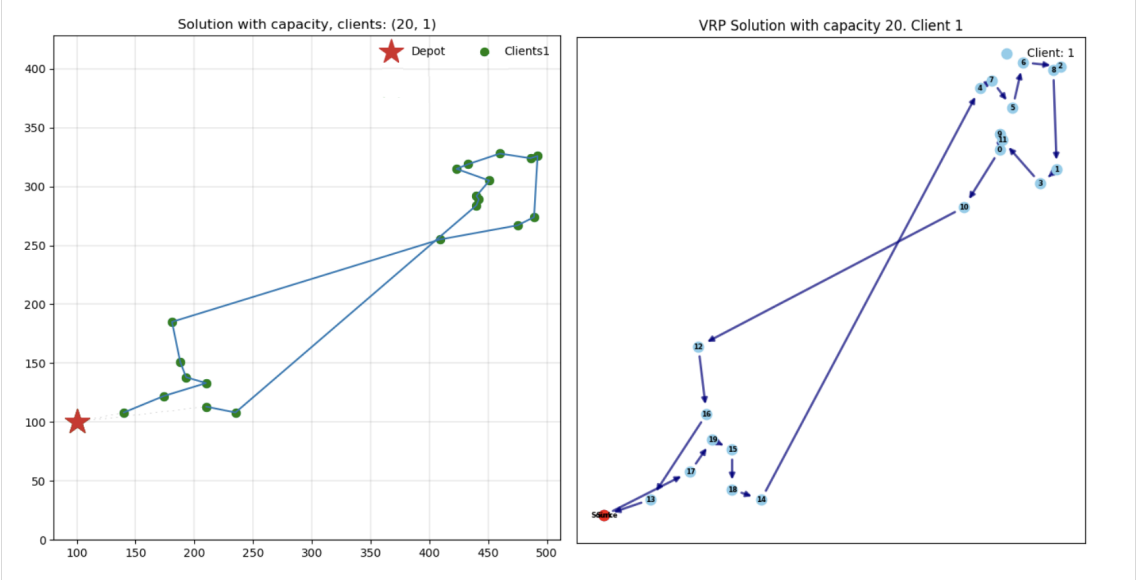

Figure 3.3: Route for customer 1 after 2 hour runtime. PyVRP (left) VRPy (right)

As both algorithms face the same limitations, the chosen algorithm was based on the stopping criteria. After evaluating the advantages and limitations of the algorithms, the final decision is to use PyVRP with two stopping criteria: no improvement after 500 iterations and a time limit of one minute.

3.2 Feature generation

The instances used in this study only consists of the delivery locations required by each company. Since the locations of these shipments alone are insufficient to gain insights into the shipment characteristics of the companies to know which companies are compatible, additional features must be generated. This section focuses on creating features aimed at identifying compatible companies and identifying optimal collaborative partners. Many of these features draw inspiration from the insights gathered during the literature review outlined in chapter 2.

3.2.1 Clustering

As discussed in 2.1.3, clustering can significantly enhance the efficiency of solving VRPs, making it better for scaling. Therefore, in this study, clustering will be employed to identify

more compatible companies for collaboration, considering the number and size of shared clusters, as well as estimating the cost of the routes between these clusters.

3.2.1.1 K-means clustering

A new feature utilizing K-means clustering was developed to determine whether companies share numerous deliveries within the same geographical areas. If two or more companies primarily deliver to the same regions, they are deemed compatible. This capability is advantageous in reducing logistical expenses, primarily addressing scenarios where multiple companies independently transport half-empty trucks to the same destination. Through collaboration, they can combine their deliveries into one fully loaded truck for that specific region, thereby eliminating the need for an additional vehicle and reducing logistic costs. Additionally, this collaborative approach decreases traffic congestion and environmental harm by reducing the number of vehicles on the roads.

3.2.1.2 Silhouette score

When utilizing K-means, the algorithm requires predefining the number of clusters used. The performance of the algorithm is highly dependant on the number of clusters and therefore to find the optimal number of clusters the silhouette score is utilized [21]. The silhouette score is used to measure the goodness of the clustering. The score ranges from -1 to 1. A value of 1 means that the clusters are well apart from each other and very clearly distinguished. Alternatively, a value of -1 means that the clusters are assigned in the wrong way and a value of 0 means that the clusters are indifferent and the distance between the clusters is not significant. The silhouette score is calculated by taking the mean of the silhouette coefficient for each sample. The silhouette coefficient measures how well a sample is clustered with samples that are similar to themselves. For each sample the silhouette coefficient is calculated with the following formula:

$$\text{silhouette score} = \frac{b - a}{max(a, b)},$$

where a is the average intra-cluster distance (average distance between each point in the cluster) and b is the mean nearest-cluster distance (the average distance between all clusters).

Boundary nodes When clustering the nodes, the number of clusters is determined by the silhouette score. However, the clustering process does not consider the ownership of the deliveries within the clusters. Consequently, there may be deliveries that fall on the boundary of two clusters and could belong to either one. A delivery lies on the boundary of two clusters if it is equally close to the centroids of both clusters. For instance, let's consider a node that belongs to company A. This node is on the boundary of two clusters, 1 and 2, and the K-means algorithm assigns it to cluster 2. If cluster 1 has many deliveries from company A while cluster 2 has none apart from this boundary delivery, placing it in cluster 2 could skew the results of other features, such as the **VRP between clusters** in section 3.2.2 and **boxing method** in section 3.2.5. To address this issue, all nodes that fall close to the boundaries of the clusters are examined, and they are placed in the cluster where they share ownership with other nodes. Here, "close" is defined as 5% of the maximum distance between any two deliveries in the instance, where 5% was chosen based on numerical experiments. Figure 3.4 illustrates this.

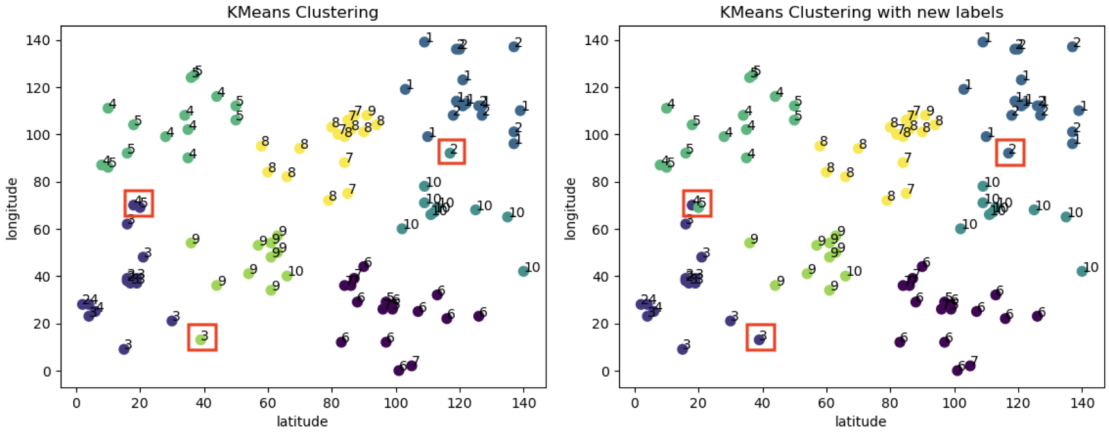

Figure 3.4: This example illustrates the boundary nodes. The nodes within the red squares lie on the boundary of two clusters. The figure on the left shows the original clustering, while the figure on the right shows when the boundary nodes have been reassigned.

3.2.1.3 Minimum/maximum number of clusters

Many of the features that have been created depend on the clustering of the orders. This is either comparing the compatibility of the companies in terms of how many orders they share in each cluster, or using the centroids of the clusters to estimate the gains or characteristics of the route for the orders. Therefore, creating the clusters is very important, specially in

terms of how many orders should be in each cluster. There should be a minimum and a maximum number of orders in each cluster.

In this study these numbers were set by how many orders should be on average in each cluster. This means that the total number of deliveries within an instance were considered where the minimum number of clusters was set to $\frac{\text{total number of deliveries}}{15}$ and the maximum number of clusters was set to $\frac{\text{total number of deliveries}}{5}$, meaning that on average there are around five to fifteen deliveries within a cluster, depending on the best silhouette score for each instance. In this study the initial thought was to start with small clusters and seeing how much could be gained. This decision was also made keeping in mind that in cases where deliveries were very spread out, with no clear clustering of deliveries, the highest silhouette score often resulted in very few clusters. This is illustrated in figure 3.5, where the deliveries cover most of the total area, creating no clear clusters. This means that, in this case, when comparing different number of clusters ranging from two and higher, the best silhouette score resulted in only four clusters. However, with only four clusters, the clustering-based features become less effective at identifying patterns within the data. This is primarily because these features perform best when the distance between cluster centroids is greater than the radius of the clusters.

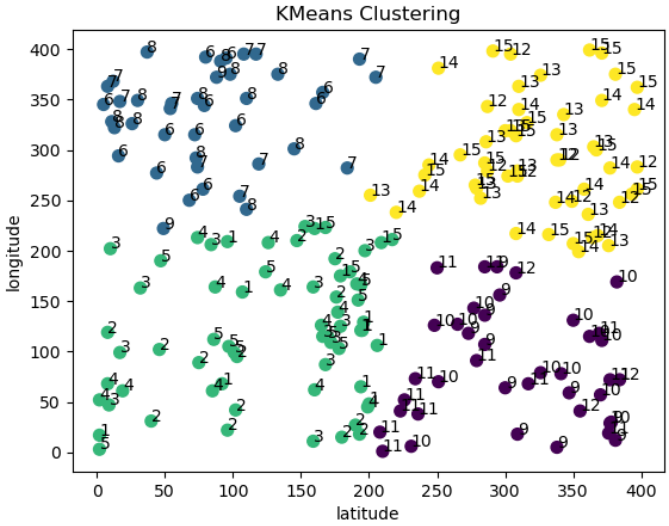

Figure 3.5: This example illustrates the clustering of the orders, here 200 orders in total. The total number of clusters is based on the highest silhouette score. The highest score is associated with only four clusters.

This includes the features **VRP between clusters** further discussed in section 3.2.2, where the routes get way too underestimated with very large clusters. This is due to the

fact that if there are very few clusters with a lot of deliveries, the radii of these clusters consequently become very large. In the estimated VRP between clusters, the total distance is estimated by only visiting the centroids of the clusters the deliveries belong to. For this estimation to represent the real distance, the radii of the clusters can not be too large. The radii need to be small enough such that all deliveries within the cluster are close enough to the centroid that the extra distance needed to be travelled to reach the delivery can be neglected. This is illustrated in figure 3.6. The figure shows the **VRP between clusters** where the clusters have been formed as illustrated in figure 3.5. This is the case where company 1 and 4 collaborate. All deliveries from both companies land in the same cluster, including the depot being very close to the centroid of that cluster. The red star represents the depot while the light green dot represents the centroid of the cluster. The estimated distance to visit all the deliveries for the collaboration is then estimated by driving twice (the capacity of the trucks in this case are 20, with the total demand of 40 within the cluster) from the depot to the centroid. This distance is way shorter than the total distance needed to be driven to visit all deliveries and hence highly underestimating the total travel distance. Therefore, to be able to have as close estimate as possible, while still being able to lower the complexity of the problem, small clusters will be used. Clusters of average size five to fifteen orders per cluster.

3.2.1.4 Similarity

To determine if two or more companies are compatible based on the K-means clustering, it is important to know which companies share deliveries in the same clusters and hence, to know, which companies are similar in that matter. To do so it is important to find a good measure that can well determine how similar the companies are.

Dot product One way to determine if two or more companies are compatible is calculating the dot product between the vectors where each vector represents how the deliveries of a company are distributed between the clusters. The dot product is calculated as follows:

$$\mathbf{a} \cdot \mathbf{b} = \sum_{i=1}^{n} a_1 b_1 + a_2 b_2 + a_3 b_3 + ... + a_n b_n,$$

where $\mathbf{a}$ and $\mathbf{b}$ are two vectors or length n. When using the dot product, both direction and magnitude matter.

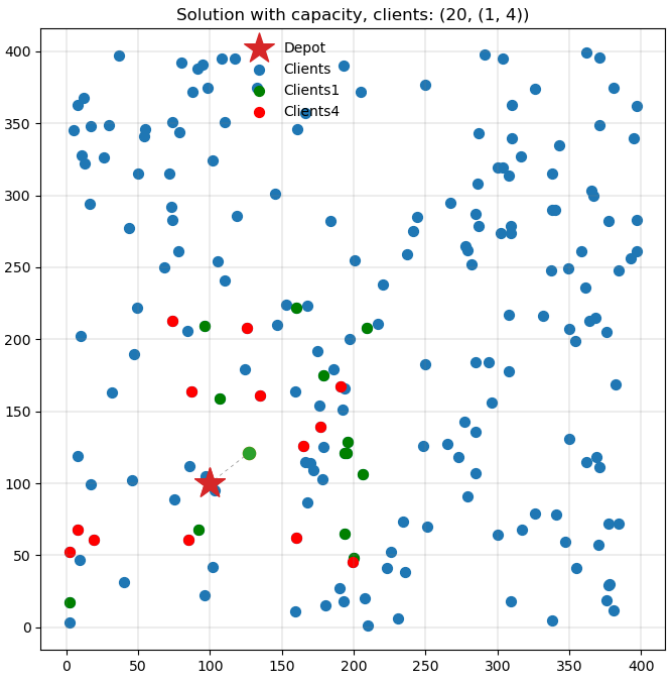

Figure 3.6: This example illustrates the VRP between clusters, based on the clusters in figure 3.5. This is for clients 1 (green) and 4 (red). This shows two routes to the same centroid (light green and orange), where all orders belong to the same cluster.

Cosine similarity Another way to determine if two or more companies are compatible is utilizing the cosine similarity measure. The cosine similarity is a similarity metric that measures how similar two vectors are to each other. The cosine similarity is calculated in the following way:

$$\cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{||\mathbf{A}|| \cdot ||\mathbf{B}||} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2} \sqrt{\sum_{i=1}^{n} B_i^2}},$$

where **A** and **B** are the vectors being compared. The value of the cosine similarity ranges from -1 to 1, where 1 means that the vectors are identical and -1 means they are absolutely opposite. A value of 0 means that the vectors are orthogonal to each other. When using the cosine similarity, only the direction matters. The magnitude of the vectors does not have an effect on the result.

Similarity for three companies The two metrics discussed earlier focus solely on comparing pairs of companies. However, this study seeks to extend this analysis to identify similarities among more than just two companies. To achieve this, an alternative version is employed when comparing multiple companies simultaneously. In this approach, the

similarity between all possible combinations of companies is computed, and then an average of the two highest ones are used. Lets consider three companies: (A,B,C). If (A,C) and (B,C) are compatible then all three are considered compatible, even though (A,B) are not. Figure 3.7 shows an example of this. In this scenario, companies 1 and 2, as well as companies 1 and 3, share all deliveries within the same cluster. This suggests that these three companies could be a good fit for cooperation when considering a group of three. However, since companies 2 and 3 do not have any common deliveries, their similarity score would be zero. If the overall similarity score is calculated as the average of all individual similarity scores, the zero score between companies 2 and 3 would significantly lower the overall similarity score, resulting in a very low similarity for the trio.

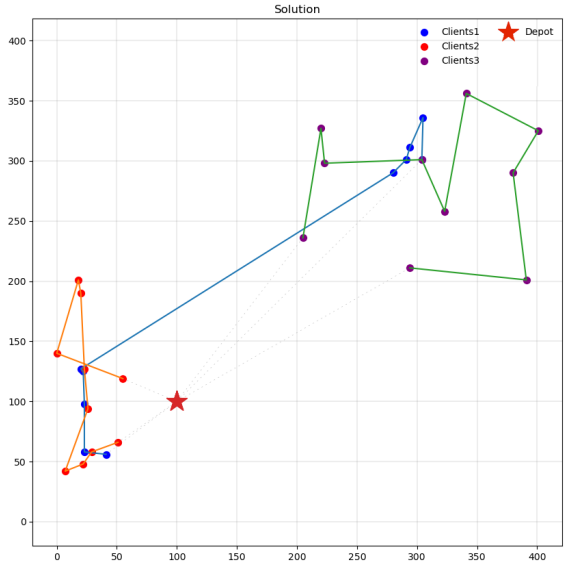

Figure 3.7: The figure shows an example of where these three companies should be considered compatible as a cooperation of three.

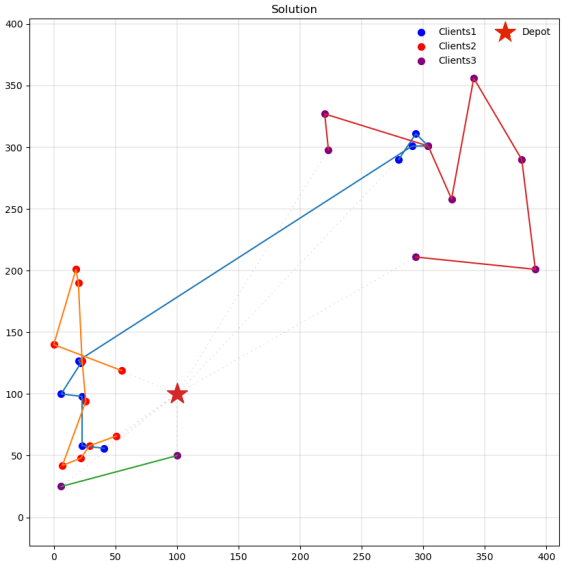

Figure 3.8: The figure shows an example of where these three companies should be considered compatible as a cooperation of three.

In comparison, figure 3.8, presents the same example, but with two of company 3's deliveries moved to the other cluster. As a result, company 3 now shares a cluster with both company 1 and company 2, making all three companies compatible. This change means that companies 2 and 3 now share clusters, leading to a high similarity score for the trio, regardless of whether the score is the average of all combinations (1,2), (1,3), and (2,3) or the highest two out of these three. However, these two scenarios should be nearly equally compatible since both facilitate effective cooperation among the three companies.

Therefore, by considering the compatibility as the average of the highest two combinations, this issue is resolved.

As the number of cooperating companies increases, this metric becomes more difficult to apply and yields less accurate results. Therefore, it will only be used for up to three cooperating companies.

3.2.1.5 Delivery count compatibility

When assessing the compatibility between two companies, the K-means clustering method considers the distribution of deliveries across clusters. This entails creating a vector for each company, where each index corresponds to a cluster and the value represents the percentage of deliveries to be delivered by that company to that cluster. This means that companies with significant delivery overlaps in specific regions are deemed highly compatible.

The rationale behind this feature is that if two or more companies share most of their deliveries in similar areas, they are more likely to be able to join their deliveries into fewer trucks, thereby reducing the total number of trucks required for that area. With fewer trucks needed, the total logistics cost decreases. Additionally, since these are long-term collaborations, the delivery locations are recurring and it might happen that some of those locations do not require daily deliveries. Over time, companies sharing many delivery locations in similar areas are more likely to be able to combine their deliveries into the same trucks more frequently, thereby minimizing the number of trucks needed. Consequently, they are more likely to benefit significantly from cooperation.

To assess this compatibility, a vector is generated for each company. Each index in the vector corresponds to a cluster. If the company has deliveries in the respective cluster, the associated value in the vector represents the proportion of deliveries the company needs to make to that cluster, relative to its total count of deliveries. For instance, if a company has a total of 100 deliveries and seven of those are allocated to cluster 3, the value at index 3 would be 0.07. To determine the compatibility between two companies the cosine similarity is calculated between the vectors associated with those companies.

3.2.1.6 Binary compatibility

However, it could also be highly advantageous for companies to collaborate in cases where one company handles numerous deliveries in a particular region while the other deals with very few deliveries in that same region. In such scenarios, the first company would invariably need to operate in that region, and for it, significant logistic costs could be saved

by the second company by joining forces with the other company for their few deliveries in that area. To capture this aspect, compatibility is evaluated using a binary variable. For each company, a vector is constructed, matching the length of the number of clusters. Each index in the vector corresponds to a cluster. If the company has deliveries in the respective cluster, the associated value in the vector is set to 1; otherwise, it remains 0. To determine the compatibility between two companies the cosine similarity is calculated between these binary vectors.

3.2.1.7 DBSCAN clustering

The DBSCAN (Density-Based Spatial Clustering of Applications with Noise) algorithm presents an alternative clustering approach. As mentioned in Section 2.1.3, DBSCAN is frequently used for pattern discovery in large location datasets due to its ability to identify clusters of arbitrary shapes and handle noise in the data. However, Li et al. [17] noted that a significant shortcoming of the DBSCAN clustering algorithm is its difficulty in clustering non-uniform location data. The DBSCAN clustering method identifies clusters by expanding from core samples of high density, making it particularly suitable for datasets with clusters of similar density [21]. However, in this study, the objective of employing this feature is to complement the K-means clustering algorithm. Unlike K-means, which clusters all samples regardless of their density or distribution, DBSCAN offers the advantage of not requiring a predefined number of clusters; rather, it only uses a predefined distance metric. This distance metric is utilized to group deliveries, with items clustered together only if their distance meets or is less than the specified metric. The rationale behind integrating this feature is to capture companies with densely clustered deliveries, indicating a potential for additional gains through collaboration.

3.2.2 VRP between clusters

As discussed in 2.1.3, Dondo et al. [10] introduced a three-step heuristic clustering approach for solving VRPs. The first step involves clustering the orders, the second step is to solve a VRP between the clusters, and the third step is to solve the VRPs within the clusters. This approach enables larger instances to be solved more efficiently, making it more scalable. Inspired by this approach, particularly phase two of the algorithm, a new feature is created. The rationale behind the feature is that for each potential collaboration of companies, a VRP is solved. In this VRP, the nodes to be visited are the centroids of the clusters that need to be visited, and the demand is the total demand of all orders that need to be

fulfilled within that cluster for that collaboration. The rationale behind this approach is to provide an approximate idea of the total logistic cost of the partnership. However, it should be noted that the centroids may not always be perfect estimations, as the clusters may cover a large region, resulting in centroids that could overestimate or underestimate the distances. However, as mentioned in section 3.2.1.3, the minimum and the maximum number of deliveries within a cluster are set to five and fifteen, such that the clusters do not get too large, and hence, the estimation being very far from the true value. Another way to address this is by incorporating the silhouette score from 3.2.1.2 as a feature. This may help capture the relationship between the density of the clusters and how accurately the centroids are estimated.

By applying the *VRP between clusters* instead of applying the vehicle routing problem to individual deliveries, the complexity of the problem is significantly reduced. This is because each cluster typically contains five to fifteen deliveries, and companies often have multiple deliveries in the same areas. As a result, the numerous deliveries within a cluster that previously needed individual attention are now represented as a single delivery.

This is illustrated in figures 3.9 and 3.10. Figure 3.9 shows the best found routes when companies 1 and 4 cooperate (with deliveries from other companies omitted from the graph for clarity). In comparison figure 3.10 shows the best found routes after clustering the deliveries. In this case, only the centroids of the clusters are visited, reducing the VRP from 80 nodes to just 16.

These figures also highlight the main shortcoming in the *VRP between clusters* when estimating routes, as indicated by the red boxes. In the *VRP between clusters* feature, all deliveries within a cluster are combined into a single large delivery, with the total demand being the sum of all individual demands. Consequently, these deliveries are assumed to be handled by a single truck. However, as illustrated in figure 3.9, this assumption does not always hold. The red boxes indicate cases where all deliveries belong to the same cluster, yet the best solution, resulting in the lowest cost, appears to involve splitting these deliveries across two or more routes.

3.2.3 VRP between clusters: allowing for split deliveries

To address the aforementioned limitation of the previous feature, a new approach has been introduced. Instead of combining all deliveries within a cluster into a single large delivery, all deliveries are relocated to the cluster centroid. This adjustment allows each delivery to remain independent and potentially be handled by different trucks. The goal is that by positioning all deliveries at the centroids, the VRP algorithm can find the best solution

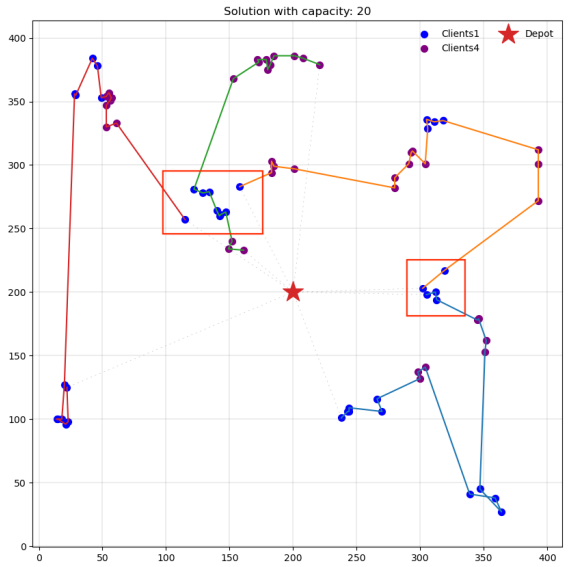

Figure 3.9: The figure shows the VRP between all deliveries when companies 1 and 4 are cooperating. (Total cost: 3172)

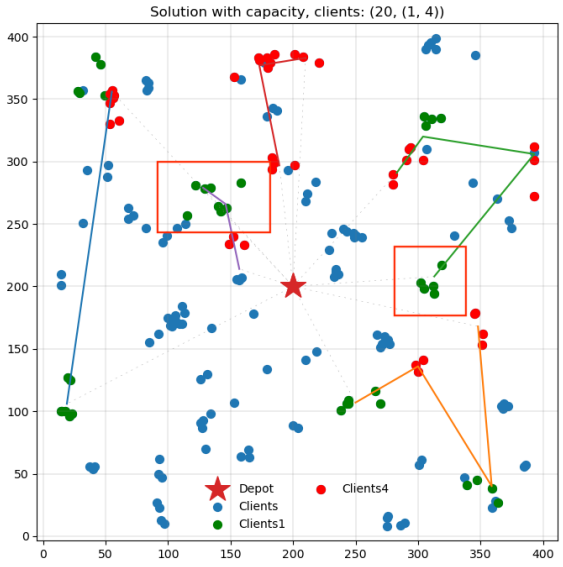

Figure 3.10: The figure shows the VRP between all cluster centroids where the collaboration of companies 1 and 4 have deliveries. (Total cost: 2926)

quicker. While this approach resolves the primary issue of the previous feature, it retains the same number of deliveries as applying the vehicle routing problem to each individual delivery. Therefore, the overall solving time of the instance may not necessarily improve. This is further explored in section 4.1.2, where its effectiveness is considered.

Figure 3.11 shows the same instance as figures 3.9 and 3.10, but with split deliveries. The estimated routes in this figure closely resemble the real routes shown in figure 3.9. However, when considering the total cost of all scenarios, the VRP without split deliveries appears to be closer to the real cost. It should be noted, though, that the estimation will often be lower than the actual value because routes within clusters are ignored. Therefore, even if the other cost value seems closer to the real value, it does not necessarily indicate a better estimate. The goal is not to have an estimation as close as possible to the real value, but to ensure that all estimations for all collaborations within an instance are representative of their real values in a relative sense. This means that an estimated values should be lower than another value if the real value is lower than another value and higher if the real value is higher. Consequently, based on the figures and routes, the method allowing for split deliveries appears to be better for our purpose, despite the total cost being further from the real cost.

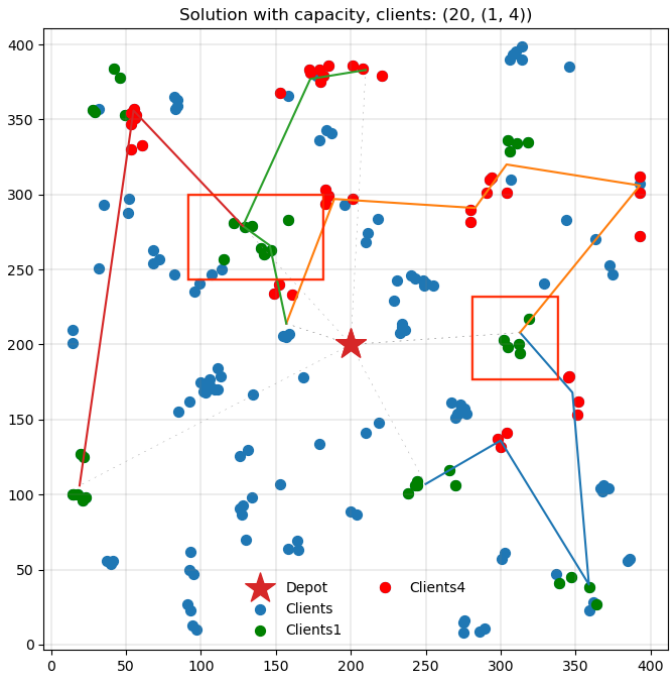

Figure 3.11: The figure shows the VRP between all cluster centroids where the collaboration of companies 1 and 4 have deliveries, allowing for split deliveries. (Total cost: 2692)

3.2.4 VRP between clusters: before and after cooperation

The *VRP between clusters* feature discussed in subsection 3.2.2 provides an approximation of the total logistic cost of the partnership, however, it does not capture if the estimated cost is lower or higher than the original cost of the companies within the partnership working on their own. The difference between the total cost of the companies working in collaboration and the total cost of them all working independently, define the benefits that can be obtained from cooperation. The gains are what we aim to figure out and for that reason the *VRP between clusters* is not only computed for every possible partnership, but also for every company working on their own. This way it is possible to estimate the benefits of the partnership, where the estimated total cost of the partnership is compared to the total estimated cost of each company within the partnership working on their own. Since the objective of this study is to identify the partners that would benefit the most from cooperation, this feature provides significantly more insight than merely considering the total cost of the routes when the companies are cooperating.

Including cost of initiating and maintaining cooperation Similarly to the previously mentioned approach, this feature computes the estimated benefits of cooperation,

however, now the fixed cost of starting and maintaining the cooperation is taken into account. This allows for more detailed cost estimation where both the routing cost as well as all additional cost is taken into account. This also estimates the real total gains better. Additionally, this enables any user of the algorithm to specify their own estimated total cost of cooperation. It also allows for different costs for each potential collaboration, helping the algorithm generalize better to instances with unseen fixed costs. In this study, the algorithm used to predict the potential benefits of cooperation is trained on multiple benchmark instances, each including a fixed cost of cooperation that increases linearly with the number of cooperative companies. However, this linear relationship is rarely observed in practice, as the total cost of initiating and maintaining cooperation varies significantly between different situations. Consequently, the fixed cost of cooperation is unlikely to be included in the training data. By incorporating the fixed cost into the *VRP between clusters*, the algorithm should better generalize to unseen instances where the total observed fixed cost has not been encountered before.

3.2.5 Boxing method

The boxing method is employed to capture compatibility along the route. This is because there may be instances where companies exhibit high compatibility without sharing any clusters. For instance, consider two companies: the first has deliveries in two clusters, while the second has none in those clusters. However, the second company's deliveries are situated along the route from the depot to either of the clusters, between the clusters, or from one cluster back to the depot. In this scenario, if these two companies were to collaborate, they would share the same route and should thus be considered highly compatible. This method draws inspiration from Creemers et al. [3], who utilized a bounding-box approach to identify compatible companies. In their methodology, they initially compiled a list of potential cooperative partners based on similar latitude coordinates, followed by the application of the bounding-box method to confirm compatibility.

Figures 3.12 and 3.13 illustrate such a case. Figure 3.12 shows the route if company 1 operates alone, while figure 3.13 shows the route if companies 1 and 2 collaborate. The figures show that the route remains largely unchanged whether company 1 works alone or if both companies work in collaboration. This indicates that their combined cost would be similar to the original cost incurred by company 1 alone. Consequently, the total cost of cooperation is nearly equal to what company 1 originally paid, suggesting that both companies can benefit significantly from collaborating. However, based on the delivery locations, it is highly unlikely that the two companies share any clusters. This means

that the clustering compatibility features, such as K-means clustering discussed in section 3.2.1.1 and DBSCAN clustering discussed in section 3.2.1.7, would indicate that the two companies are not compatible, as their clustering similarity scores would be zero or close to zero, despite these companies being quite compatible.

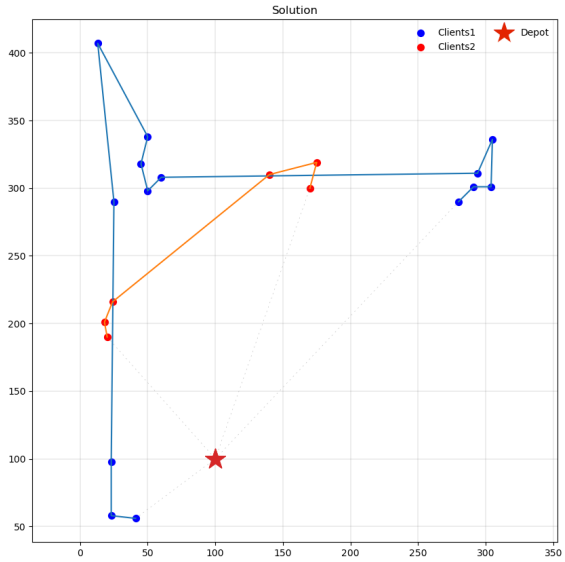

Figure 3.12: The figure shows the route if both companies work on their own.

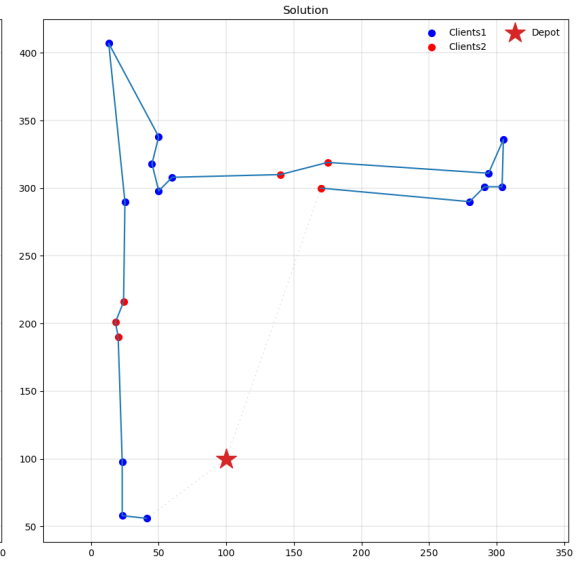

Figure 3.13: The figure shows the route if the two companies start a collaboration.

To capture this, boxes are created for each company between all clusters where the company has deliveries and the depot. These boxes are then examined to see if they contain any deliveries from other companies. Figures 3.14 and 3.15 illustrate these boxes. In the figures, the blue points represent the deliveries, numbered based on their owner, and the green star represents the depot. Figure 3.14 shows the boxes for company 1. Here, boxes are created between any two clusters where company 1 has deliveries and the depot. These boxes capture deliveries from company 2 that lie along the route between clusters or between a cluster and the depot. Deliveries within the red boxes are considered potential collaborative partners, as company 1 is likely to pass by them anyway. Similarly, figure 3.15 illustrates this from company 2's perspective. However, as shown in the figure, company 1 has no deliveries that lie on the route for company 2. To take that into account, this feature only considers the highest "on-the-way" compatibility score, which in this case is based on company 1's perspective.

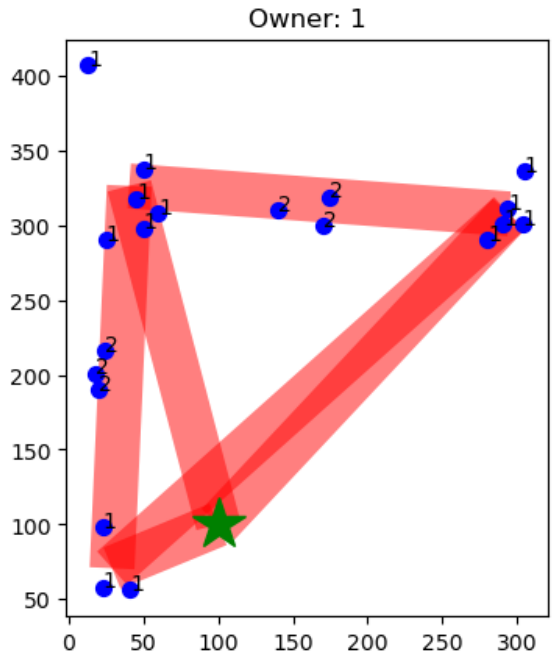

Figure 3.14: The figure shows all boxes between all cluster owner 1 has deliveries in and the depot.

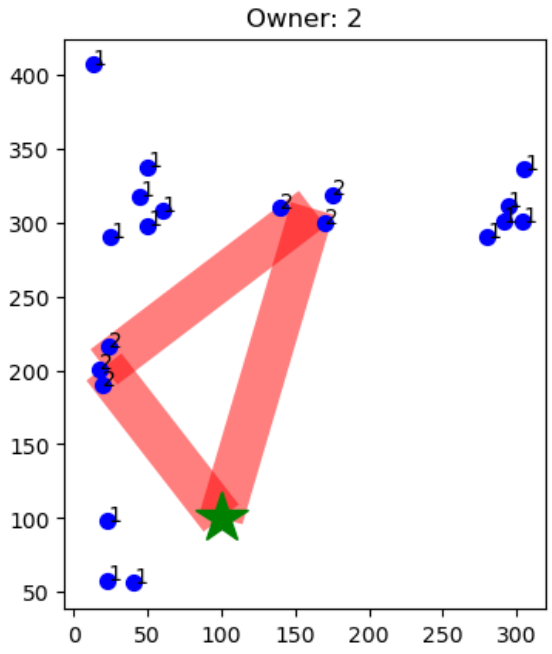

Figure 3.15: The figure shows all boxes between all cluster owner 2 has deliveries in and the depot.

On the way compatibility: binary In this study, for each company, a box is created between all clusters the company has deliveries in and the depot. This process returns a list of other companies that may potentially be on the route between any two clusters or be on the way to or from the depot. Compatibility between two companies is determined based on whether there are any deliveries along the other company's route. A compatibility score of 1 is assigned if there are deliveries on the route, otherwise, it remains 0.

In the example shown in figures 3.14 and 3.15, company 2 would receive a score of 1 for company 1 because company 2 has deliveries along company 1's route. Conversely, company 1 would receive a score of 0 for company 2, as it has no deliveries along company 2's route. For each collaboration, the highest score is considered. Therefore, the collaboration between companies 1 and 2 would receive a score of $\max(1, 0) = 1$.

On the way compatibility: order count However, a binary score might oversimplify the compatibility between companies. The number of deliveries on the way could provide a better indication of compatibility. If most or all of company A's deliveries are along the route for company B, particularly if company A's other deliveries are far from those on

the way for company B, it would significantly reduce company A's costs of cooperating with company B, while adding minimal additional cost for company B. To address this, for two companies A and B, we calculate the percentage of deliveries that A has on B's route and vice versa. The maximum of these two percentages is taken as the compatibility score. For example, if company A has 5 out of 10 deliveries that fall on the route between the clusters of B, the compatibility score would be 0.5. However, if B has 7 out of 10 deliveries on the route between A's clusters, the compatibility score for cooperation (A,B) would be updated to 0.7. A maximum score of 1 indicates that one company has all their deliveries on the route of another, while a minimum score of 0 means none are on the route.

It should be noted that while this feature has the potential to identify deliveries that fall along the route of another company, its effectiveness diminishes in larger instances where the companies have deliveries spread across multiple clusters located over a wide area. In such cases, each company would have boxes between every two clusters and the depot, potentially covering a significant portion of the total area. This could mistakenly identify almost all other companies as compatible, even though they may not actually be compatible, as these boxes only indicate possible routes rather than routes that will necessarily be traveled. Therefore, this feature is more useful in scenarios with few deliveries per company or where deliveries are very clustered in only a few clusters. An example could be multiple companies operating in the Netherlands, where each company delivers to only a few cities. This feature could potentially capture cities that lie between the other cities the truck needs to travel to. However, it becomes less useful in cases where all companies are located within Amsterdam and have deliveries spread throughout the whole city.

3.2.6 Truck utilization

When trying to estimate the gains of collaboration, the truck capacity plays a crucial role. If the number of deliveries in a cluster slightly exceeds a truck's capacity, it will result in the collaboration having to invest in an extra truck, which leads to low or even no gains. For the same reason if one truck can handle all the deliveries in a cluster it can lead to very high gains.

Truck utilization per cluster To capture this property a feature was created to calculate the number of trucks needed to handle all deliveries in a cluster. This means that for each cluster the utilization of the collaboration is calculated and added as a feature. However, since the clusters can vary between instances, this approach might not be very

practical. To improve this, a feature was created to calculate the **count of clusters needed more than a single truck**. If a collaboration between two companies results in many clusters requiring more than one truck, it may not yield significant benefits. In such cases, each cluster in the collaboration may need to be visited by two trucks, resulting in similar costs as before the collaboration. Therefore, collaborations with many clusters exceeding full utilization (1 truck per cluster) should receive a lower compatibility score.

Total truck utilization Another feature was created to calculate the total truck utilization required to handle all deliveries in every cluster for the collaboration. This feature aims to determine the number of trucks needed and to quantify how many trucks are saved through collaboration. If significantly fewer trucks are used in collaboration compared to when the companies worked independently, the collaboration should be considered more compatible.

Trucks saved based on clusters This feature aims to capture how many trucks can be saved by cooperation, taking into account how many trucks the companies needed to send to each cluster when they are working independently, compared to how many they need to send when cooperating. Lets consider two companies, A and B. Both have deliveries in cluster 1. However with two trucks of capacity 10, company A has 12 deliveries in that cluster and company B has 13. By working on their own both have to send two trucks to that clusters, compared to working together where they only need to send three combined. This saves a truck, which might be a good indicator of potential gains. This is calculated for each cluster the companies of the collaboration have deliveries in, determining the total number of trucks that can be saved through collaboration while considering the locations of the deliveries.

3.2.7 Shared routes

As previously mentioned, solving a VRP is very computationally heavy, making it impractical to solve every possible combination of collaborative partners to determine the most beneficial ones. However, it is feasible to combine all deliveries from all companies into a single instance and solve it as a whole. This allows us to identify companies that frequently share routes, if treated as one, which can indicate potential candidates for cooperation. Figure 3.16 (left) shows an instance where a VRP has been solved for all deliveries from all companies, with each truck having a capacity of 20. This figure illustrates which companies

would share routes if all deliveries were considered, along with the total number of shared routes.

3.2.7.1 Percentage route sharing

When looking at the shared routes, the route count alone may be misleading. That is, simply considering the number of shared routes. This is because some companies may have numerous deliveries or smaller trucks, resulting in more routes, while others may have fewer deliveries or larger trucks, resulting in fewer routes. Therefore, the significance of the number of shared routes depends on the total number of routes. To address this, the route count is scaled as a percentage of the total number of routes shared. This provides a better indication of whether companies share many routes and ensures scalability across different instances.

3.2.7.2 Percentage route sharing: considering delivery count

When only considering the percentage of shared routes, it overlooks the number of deliveries handled on those routes. For example, if all deliveries from all companies can fit into one truck, the route percentage would be 100% for all combinations of companies, regardless of their actual compatibility. This can be quite misleading and therefore it is very important to capture the capacity of the trucks in combination with the number of deliveries that need to be delivered. One way to do so is to multiply the percentage of the shared route by $\frac{\#deliveries}{truck\ capacity}$. This means that when trucks are very small compared to the number of deliveries, the value of the route sharing feature becomes higher, indicating that in those cases the number of routes feature is more informative and should be higher.

3.2.7.3 Clustered route sharing

When constructing the routes for all the deliveries combined, the ownership of the deliveries is not considered. This can lead to situations where a company, say company A, has all its deliveries on route 1, except for one delivery on route 2. However, the delivery on route 2 is very close to the ones on route 1. This would result in an overestimation of how many routes are shared or not shared. To prevent this from skewing the final results, the closely clustered deliveries of each owner are combined and treated as a single order, with the demand being the sum of all the individual deliveries' demands. This ensures that clustered deliveries for the same owner are all included in the same route.

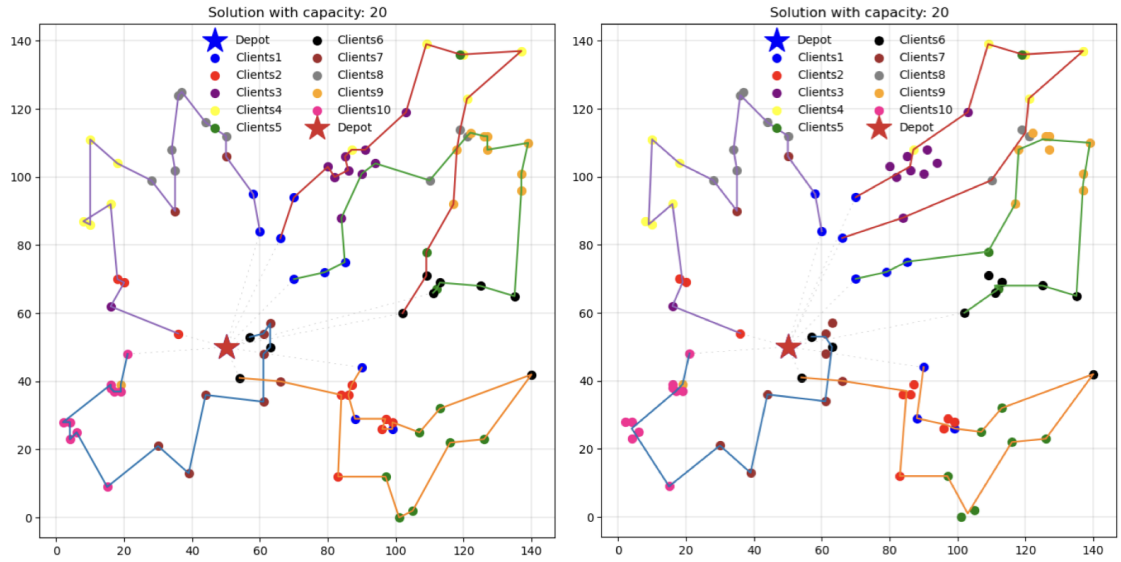

Figure 3.16: Shows the routes that are constructed before clustering (left) compared to after clustering (right)

This is illustrated in figure 3.16. The figure on the left shows the routes before clustering, while the figure on the right shows the routes, when the deliveries were clustered before the routes were constructed. It can be seen that before clustering, the deliveries from company 3 (shown with purple deliveries) belong to both the red and the green route, even though these deliveries are all very closely located and could easily belong to the same truck. In the right figure, after clustering, all these deliveries belong to the red route. This illustrates that before clustering, company 3 shared two out of five routes with company 6 (shown with black deliveries). However, after clustering, these companies no longer share any routes, indicating a more accurate representation since they do not share many deliveries in similar areas.

3.2.8 Average distance to depot

This feature aims to identify companies with significant potential to benefit from collaboration. Essentially, it examines scenarios where a company's usual operational route is very expensive in terms of the distances the trucks need to travel. In such cases, collaboration is more advantageous compared to situations where deliveries are close to the depot. The rationale behind this lies in the likelihood that expensive routes entail deliveries spread far apart or distant from the depot. When multiple companies share this characteristic, there

exists the opportunity to maximize gains by combining distant deliveries onto the same truck.

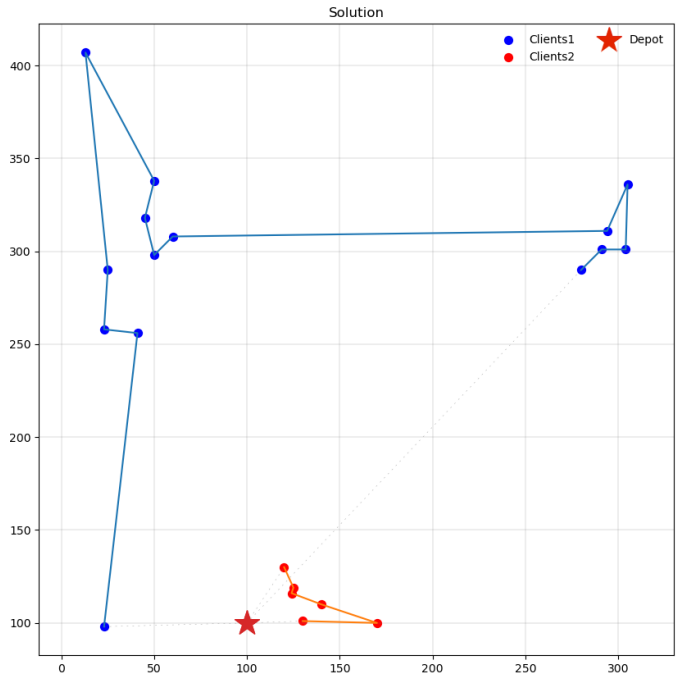

Figure 3.17: In the figure, two scenarios are depicted: one where the company has deliveries located far from the depot, and another where deliveries are located very close.

Figure 3.17 illustrates this contrast. Company 2 (shown with red deliveries) has all deliveries located closely together and near the depot, resulting in a relatively inexpensive route. In contrast, company 1 (shown with blue deliveries) faces higher costs due to deliveries located far from the depot and spread across a larger area. Moreover, company 1 has isolated deliveries, one in the lower left corner and another in the upper left corner, requiring detours for the truck. This presents an opportunity for potential gains if there exists another company facing similar challenges with expensive routes and deliveries in comparable areas.

3.2.9 Company characteristics

As stated by Cuervo et al. [8], the characteristics of companies are crucial in determining compatible partners for collaboration. Therefore, certain useful characteristics are included as features.

Collaboration size Since this study seeks to identify potential partnerships involving two or more companies, the number of companies within the collaboration can provide valuable insight into the benefits of cooperation.

Collaboration cost In this study it is assumed that there are costs associated with forming partnerships. This reflects real-life scenarios where both initiating and sustaining collaborations can result in significant costs. To account for this, collaboration costs are included as a feature in the algorithm.

Total number of deliveries As discussed in section 2.1.4, the total number of deliveries significantly impacts the profit generated by the collaboration. The more deliveries a company has, the greater the potential gains when collaborating with a partner. Therefore, this will also be included as a feature.

Truck capacity and average order size As mentioned earlier in section 3.2.6, the capacity of trucks plays a crucial role in identifying potential partnerships. This is also in regard to the findings of Cuervo et al. [8], who emphasized the significance of average order size as the most influential factor. Companies with varying order sizes that can efficiently fit together in a truck stand to benefit the most. However, in this study, all orders are assumed to be of the same size, making it unnecessary to include average order size as a feature. For that reason only the truck capacity is added as a feature.

Total cost before collaboration Similarly to the concept of **average distance to depot** discussed in section 3.2.8, the total cost before collaboration examines scenarios where a company's usual operational route is very expensive. This identifies companies that might gain a lot from collaboration. Conversely, companies with already low operational costs might be less suitable for collaboration, as their routes may already be optimized with less room for improvement through collaboration.

3.3 Machine learning models

The machine learning models that will be utilized in this study will be used to predict the expected gains that can be obtained for a collaboration. As a result, all selected models will be regression models. These predicted gains can then be ranked from highest to lowest, allowing identification of the collaboration with the highest potential gains as well as being able to obtain the combination of collaborative partners that result in the total maximised

gains. The chosen models will range from simple, easily interpretable models, such as decision trees, to more complex models, like XGBoost regression models.

3.3.1 Decision tree

The first model that will be used is a decision tree. A decision tree is a supervised machine learning algorithm that can be used for both classification and regression tasks [24]. A decision tree uses a tree structure to make predictions over a desired target variable. Compared to even simpler models like linear regression, decision trees allow for more complex relationships between the variables where they do not assume linearity. The benefit of using a decision tree is that these models are relatively simple, making them easier to interpret. However, their simplicity might not allow for more complex relationships between the features. This model will serve as a baseline model.

Hyperparameter tuning For optimal performance, hyperparameter tuning is important. Tuning these parameters can significantly enhance the model's accuracy. The following list describes the parameters chosen for tuning the model.

- Max depth: Controls the maximum number of levels in the tree. Limiting the depth can help prevent overfitting, especially on noisy data.
- Min samples split: This controls the minimum number of samples required to split an internal node. It determines the minimum number of samples needed to consider splitting a node. Higher values prevent the model from learning too fine details, which can help with generalization.
- Min samples leaf: This controls the minimum number of samples required to be at a leaf node. This ensures that a leaf node has at least this number of samples. Helps to smooth the model, making it less sensitive to noise.
- Max features: This controls the number of features to consider when looking for the best split. It limits the number of features that are considered for each split, which can help in reducing overfitting and improving the model's performance.
- Max leaf nodes: This controls the maximum number of leaf nodes in the tree. The purpose of this parameter is to limit the number of leaf nodes, which can control the size of the tree and help prevent overfitting.

- Min weight fraction leaf: This controls the minimum weighted fraction of the sum total of weights, of all the input samples, required to be at a leaf node.

Table 3.1 shows the parameter grid on which the parameters were tuned on and the final best values.

Parameter	Grid	Chosen value
Max depth	[None, 10, 50, 100, 200, 500]	None
Min samples split	[2, 5, 10, 20, 50, 100]	50
Min samples leaf	[1, 2, 4, 10, 20, 30, 50, 70]	10
Max features	[None, 'auto', 'sqrt', 'log2', 0.5, 0.8, 0.9]	0.8
Max leaf nodes	[None, 50, 100, 150, 200, 300]	200
Min weight fraction leaf	[0.0, 0.1, 0.2, 0.5, 0.8]	0.0

Table 3.1: The grid for hyperparameter tuning: Decision Tree.

3.3.2 Ensemble learning methods

Ensemble algorithms, are typically more complex and can capture nonlinear relationships in the data by combining multiple simpler models. Ensemble learning methods are a powerful approach where multiple models are combined to improve the overall predictive performance compared to using a single model. The diversity among the individual models ensures that each model makes different errors such that their collective decision-making process compensates for their individual weaknesses. These methods combine predictions from multiple models to make a final prediction or decision. The aggregation can be based on voting, averaging, or weighting predictions.

Ensemble methods can be categorized into two types: **bagging** and **boosting**. On the one hand bagging involves training multiple instances of the same base learning algorithm on different subsets of the training data. Random forest models fall into the category of bagging algorithm. On the other hand, boosting involves sequentially training multiple models, where each subsequent model corrects the errors of its predecessor. XG-Boost is an example of a boosting algorithm.

3.3.2.1 Random forest

A random forest algorithm is an ensemble machine learning algorithm. Random forest allows for a non linear relationship between the decision variables and the prediction. Random forest supports both classification tasks and regression tasks. However, in this

study the goal is to predict the exact gains from a collaboration of companies and therefore a regression model is appropriate. Random Forest regression model is an ensemble learning method that operates by constructing multiple decision trees during training and outputting the mean prediction of the individual trees. Each tree is trained independently on a random subset of the training data using bootstrap sampling, that is, sampling with replacement. During the construction of each tree, at each split point, a random subset of features is considered. This introduces further randomness and diversity into the model, ensuring that the trees are not highly correlated.

Compared to decision trees, random forest is less sensitive to noise and outliers. This is due to the aggregation of multiple predictions, therefore the ensemble nature of random forest makes it more robust to noise and overfitting. The aggregation of multiple predictions also reduces variance in predictions. Random forest uses random feature selection, which increases diversity and helps in achieving better generalization compared to individual decision trees.

Hyperparameter tuning Tuning the hyperparameters can significantly impact the performance and generalization ability of a Random Forest model. For that reason, the following hyperparameters were chosen and tuned:

- N-estimators: This defines the number of trees in the forest. Increasing this parameter generally improves performance until a certain point, after which the benefit diminishes or stabilizes. More trees reduce variance but increase computational cost.
- Max depth: This defines the maximum depth of each tree. Deeper trees can model more complex relationships in the data but increase the risk of overfitting.
- Min samples split: The minimum number of samples required to split an internal node. Larger values prevent the model from learning overly specific patterns, which reduces overfitting. Smaller values allow the model to capture more nuances but may lead to overfitting.
- Min samples leaf: The minimum number of samples required to be at a leaf node. Similar to min samples split, but applies to leaf nodes. Larger values can smooth the model and prevent overfitting.
- Max features: The number of features to consider when looking for the best split. Controls the randomness of feature selection for each tree. Lower values increase diversity among trees but may decrease predictive accuracy.

Table 3.2 shows the parameter grid on which the parameters were tuned on and the final best values.

Parameter	Grid	Chosen value
Max depth	[None, 10, 20, 30, 50]	None
Min samples split	[2, 5, 10, 20]	2
Min samples leaf	[1, 2, 4, 8]	4
Max features	[None, 'auto', 'sqrt', 'log2', 0.5, 0.7]	0.7
Max samples	[0.5, 0.7, 1.0]	1.0
Bootstrap	[True, False]	True
N estimators	[50, 100, 150, 200, 400]	50

Table 3.2: The grid for hyperparameter tuning: Random Forest.

3.3.2.2 XGBoost

XGBoost (Extreme Gradient Boosting) is another ensemble machine learning algorithm. However, this is a boosting and not a bagging algorithm. The XGBoost is an efficient and flexible implementation of the gradient boosting framework, designed for both speed and performance, hence, being both effective and efficient. For that reason it is widely used in machine learning competitions and real-world applications. When using XGBoost, various objective values can be selected, each corresponding to a different loss function that the model aims to minimize. In this study, the chosen loss function for minimization was the absolute error.

Hyperparameter tuning As previously mentioned, tuning the hyperparameters of can significantly impact the performance and generalization ability of the model. For that reason, the following hyperparameters were chosen and tuned:

- Max depth: The maximum depth of a tree. Increasing this value makes the model more complex and increases the possibility to overfit.
- Learning rate: The step size shrinkage used in each boosting step to prevent overfitting. Smaller values make the model more robust to overfitting but require more trees to reach the same performance. The learning rate can range from 0 to 1.
- Minimum child weight: The minimum sum of instance weight (hessian) needed in a child. This parameter controls the complexity of the model. Higher values prevent

the model from learning relations that might be highly specific to the particular sample selected for a tree.

- N-estimators: The number of boosting rounds (trees) to be built. A higher number of trees can improve performance but also increase the risk of overfitting.
- Subsample: The fraction of samples to be used for building each tree. Lower values prevent overfitting but might increase variance.
- Colsample bytree: The fraction of features to be used for building each tree. Similar to subsample, lower values help prevent overfitting.

Table 3.3 shows the parameter grid on which the parameters were tuned on and the final best value.

Parameter	**Grid**	**Chosen value**
Learning rate	[0.01, 0.1, 0.2, 0.25, 0.3, 0.32, 0.35]	0.3
Max depth	[None, 3, 5, 7, 10]	3
Colsample bytree	[0.8, 0.9, 1.0]	1.0
Min child weight	[2, 3, 4, 6]	4
N estimators	[20, 30, 40, 50, 100, 200]	40
Subsample	[0.8, 0.9, 1.0]	1.0

Table 3.3: The grid for hyperparameter tuning: XGBoost.

3.4 The predicted gains

The goal of all these machine learning methods is to be able to predict the potential benefits obtained from cooperation, referred to as *the gains* in this study and defined as follows:

$$\text{The gains} = \text{Total separate cost} - (\text{Total combined cost} + \text{Fixed cost} * (n-1))$$

Where each variable is defined in the following way.

- n: Number of companies within the collaboration.
- Total separate cost: The total cost of all companies within the cooperation working on their own. This can be interpreted as the total cost before cooperation.
- Total combined cost: This is the total cost when all the companies within the cooperation collaborate.

- Fixed cost: This is the fixed cost of starting a cooperation. In this study this cost is assumed to be associated with every company added to the collaboration, hence, the multiplication by $(n-1)$.

Another way of defining the gains is by looking at the percentage savings, referred to as *the percentage gains* and defined as follows:

$$\text{The percentage gains} = \frac{\text{Total separate cost} - (\text{Total combined cost} + \text{Fixed cost} * (n-1))}{\text{Total separate cost}}$$

Using *the percentage gains* should however be used carefully, as the fixed cost associated with the collaboration is also scaled with the Total separate cost.

3.5 Iterative prediction method

The final model used in this study is an iterative machine learning method. This method utilizes all features created in the feature generation step in section 3.2 as well as a machine learning model from section 3.3. The goal of this method is to be able to predict the approximated benefits and the ranking of all possible collaborations of any size within a reasonable time-frame. To do so the first step is to only look at all possible collaborations between any two companies. Based on all created features from the feature generation step a prediction is made on the gains of that cooperation, including fixed costs. For the next step, these predictions are added as a feature to the dataset including the cooperation of any three companies. Again, predictions are made. The same is done for all possible collaborations of four companies, and so on, allowing for predictions of any cooperation of n companies.

3.5.1 Features used in the iterative method

While all features from section 3.2 are considered, some are computationally intensive. Moreover, as the number of potential company collaborations increases, the combinations grow rapidly. Therefore, selecting features at each step is crucial to minimizing computational time.

3.5.1.1 Features used for cooperation of two

For cooperation of two companies, all features described in section 3.2 are utilized.

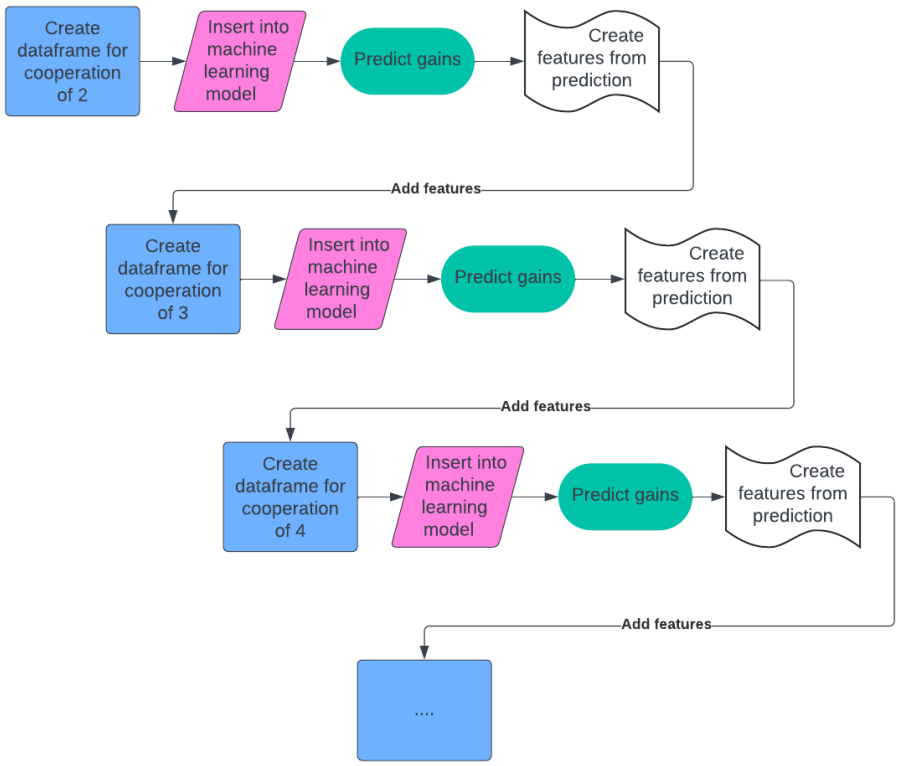

Figure 3.18: Flowchart of the iterative prediction method

3.5.1.2 Features for cooperation of three

For cooperation of three companies the number of possible collaborations grows excessively. This emphasized the need for using features that take quicker to compute. For that reason, some of the features described in section 3.2 are slower than others and will therefore not be included in the dataframe for cooperation of three. However, the features that will be incorporated are: *coalition cost*, *coalition*, *route sharing count*, *percentage route sharing*, *silhouette score*, *number of clusters*, *total number of deliveries*, *trucks saved*, *trucks saved based on clusters*, *number of clusters needing more than a truck*, *percentage of clusters needing more than a truck*, *truck capacity*, *total truck utilization*, *average distance to depot*, *total cost before collaboration* and *grand coalition cost*. Additionally, the predictions from the previous step of the iterative method will be used to create new features. All those features are based on the subcollaborations that form the grand collaboration of three companies. This means that for a collaboration of companies A, B, C we have that (A, B, C) has subcollaborations (A, B), (B, C) and (A, C). All these subcollaboration have

predicted expected gains from the previous step of the algorithm.

Subcollaborations Predicted gains for all subcollaborations are added as features. This created a total of three additional features to the dataframe of three companies.

Average of the highest two subcollaborations This feature calculates the average of the two highest subcollaborations. The rationale behind this approach is similar to the one used for determining similarity among three companies, as discussed in section 3.2.1.4.

Rank of subcollaborations This feature ranks all the collaborations within an instance and adds the rank of each sub-collaboration as separate features, resulting in three additional features.

Average the rank of the highest two subcollaborations Similar to the *average of the highest two subcollaborations*, this feature calculates the average of the two highest ranks.

Average of all subcollaborations This feature takes the average of all subcollaborations.

Highest subcollaboration This feature identifies the most promising subcollaboration among all subcollaborations within the overall collaboration.

3.5.1.3 Features for cooperation of four

For four company collaborations, the number of possible combinations increases significantly, similar to the situation with three companies. To address this, new, more efficient features are generated based on the predictions from the previous step, which focused on three company collaborations. This approach helps manage the complexity and computational demands of predicting four company collaborations.

The features used in this step are the same as those applied for cooperation of three companies. However, again, new features are generated from all subcollaborations within the larger collaboration. This means that for a collaboration (A, B, C, D) we have the subcollaborations (A, B, C), (A, C, D), (A, D, B), (B, C, D). All of the features from the previous step are utilized again, along with a few additional new features.

Average of highest three subcollaborations Similarly to the *average of highest two subcollaborations*, it takes the average over the highest three.

Average of highest three ranks Similarly to the *average of highest two ranks*, it takes the average over the highest three.

Lowest subcollaboration This feature identifies the least promising subcollaboration among all subcollaborations within the overall collaboration.

3.6 Assignment problem

The last step is to consider assignment problems. This means that when all predictions have been made then we want to assign every company to a cooperation such that the total gain is maximised.

Combination	Gains
(1, 2)	210
(2, 3)	207
(1, 4)	120
(2, 4)	74
(1, 3)	-20
(3, 4)	-35

Table 3.4: An example of the gains obtained from cooperation of two companies, with the total of four companies.

3.6.1 Linear programming

The linear programming assignment problem assigns the companies to the collaboration based on an objective function that needs to be maximised. This objective function looks at every possible combination of cooperations such that all companies are assigned to a cooperation and such that the total gains are maximised.

In the example in table 3.4, this would mean that all possible combinations of partners would be considered, as shown in table 3.5. This would lead to the last combination of $(1, 4)$ and $(2, 3)$ to be chosen, as they result in the highest overall gains.

Combinations	Total Gains
(1, 2) (3, 4)	175
(1, 3) (2, 4)	54
(1, 4) (2, 3)	327

Table 3.5: Total gains of all possible partnerships from table 3.4.

3.6.1.1 Assignment of maximum two companies

For the assignment problem, in the first case only collaborations of maximum two companies is considered. This means that the best collaboration for all involved companies consists of cooperation of two companies or individual companies.

$$\begin{aligned} \text{maximise:} \quad & \sum_{i=1}^{n} \text{gains}_i * x_i \\ \text{subject to:} \quad & \sum_{i:c\in\text{pairs}_i} x_i \leq 1, \qquad \forall c \in \{1, 2, \ldots, m\} \\ & x_i \in \{0, 1\}, \qquad \forall i \in \{1, 2, \ldots, n\} \end{aligned}$$

The first constraint makes sure that each company can appear in at most one selected pair and the second constraint is a binary constraint for the decision variables.

Variables:

- Let x_i be a binary variable that is 1 if the pair i is selected, and 0 otherwise.

Indices:

- Let $i \in \{1, 2, \ldots, n\}$ represent each pair, where n denotes the total number of collaborations.
- Let $c \in \{1, 2, \ldots, m\}$ represent each company, where m denotes the total number of companies.

Sets:

- pairs_i: is a set of companies involved in the i-th pair.
- gains_i: is the gain associated with the i-th pair.

3.6.1.2 Assignment of maximum three or more companies

The linear program for the assignments of maximum three or more companies is the same as the one described above, but indices i do not only represent the pair, but any collaboration of companies, of sizes two to four companies. Similarly, pairs_i is a set of companies involved in the i-th collaboration and gains_i is the gain associated with the i-th collaboration.

4

Results and analysis

The purpose of this chapter is to present all the results obtained from the algorithm discussed in section 3 and to provide insights into the computational time required to run the algorithm.

The sections are organized as follows: Section 4.1.1 offers an overview of the benchmark instances used in this study, which are also the instances used to train all machine learning algorithms. Section 4.1.2 compares the computational time needed to make predictions with the time required to obtain exact values for all possible collaborations. Sections 4.1.3 and 4.1.4 discuss the performance metrics used to evaluate the benchmark instances and provide the corresponding results. These sections focus on the performance in terms of predicting the best collaborations, measuring how well the algorithm identifies the highest predicted gains. In contrast, section 4.1.6 evaluates the algorithm's effectiveness in assigning each company to a collaboration that maximizes the overall gain, such that all companies benefit the most. Section 4.1.5 analyzes the feature importance of all the machine learning models used. Lastly, section 4.2 examines how well the algorithm scales with larger instances, involving either more companies, more deliveries per company, or both.

4.1 Benchmark instances

The results from this study are obtained by utilizing benchmark instances, this includes time comparison of algorithms as well as performance of all used models.

4.1.1 The instances

The experimental data utilizes the Gehring & Homberger extended Solomon's instances [14], which are commonly used examples of the Capacitated Vehicle Routing Problem with time windows (CVRPTW). Many studies on Vehicle Routing Problems (VRPs) or horizontal collaboration use this data, including Dondo et al. [10] and Cruijssen et al. [4].

Figure 4.1 shows the format of the text file that defines each of these problem instances. K indicates the maximum number of available vehicles and Q is the capacity of each vehicle. In this study K is kept large, as it is always assumed there are enough available vehicles for the problem to be solved. The Q differs between instances. 'Ready time' is the earliest time at which service may start and 'Due date' is the latest time at which service may start. However, these values are not used in this study, indicating that the there is no specific 'Ready time' nor 'Due date' for the deliveries. Similarly 'Service time' is excluded from this study.

```
<Instance name>
<empty line>
VEHICLE
NUMBER     CAPACITY
  K           Q
<empty line>
CUSTOMER
CUST NO.  XCOORD.   YCOORD.    DEMAND   READY TIME  DUE DATE   SERVICE TIME
 <empty line>
    0       x0        y1         q0         e0          l0           s0
    1       x1        y2         q1         e1          l1           s1
  ...     ...       ...        ...        ...         ...          ...
  100     x100      y100       q100       e100        l100         s100
```

Figure 4.1: The structure of the Solomons and the Gehring & Homberger instances.

The data is originally created for capacitated vehicle routing problems (CVRP) where all deliveries are assumed to belong to the same owner. However, in this study it is assumed that there are multiple companies, all with their own set of deliveries. For that reason all deliveries within the instances are split among the total chosen number of companies. This way of distributing the deliveries is inspired by the study conducted by Cruijssen et al. [4], where they divided the deliveries from a Solomon's instance equally among three companies to create an instance. Originally, the deliveries were randomly split among the companies. However, by doing so, most of the companies had deliveries spread around all possible areas, making it harder to differentiate between good collaborative partners and bad collaborative partners. If all companies have deliveries spread all over the total

area, the difference between the gains of the collaborations becomes negligible. However, if the companies have more concrete delivery locations and mostly clustered in few areas, finding a suitable partner becomes more important when maximising gains. This way the difference between good and bad partners becomes larger, emphasising the need of finding an appropriate partner. As this study focuses on finding the good partners from a pool of both good and bad partners, creating data where the difference is clearer becomes more important. For that reason, before splitting them up among the companies, K-means clustering algorithm, as described in section 3.2.1.1, is applied to the data. This is accomplished by first ordering all the deliveries based on clusters and then assigning them to each company one at a time. For instance, if the first cluster has only five deliveries but the first company is assigned six, the company will receive five clustered deliveries and one from the next cluster. This approach results in more clustered assignments compared to completely random assignments, while still incorporating some randomness to better represent real life scenarios. Figures 4.2 and 4.3 show the deliveries when they were split randomly among the companies, compared to when they were first clustered and then split among the companies. The latter ended up being the choice for this study.

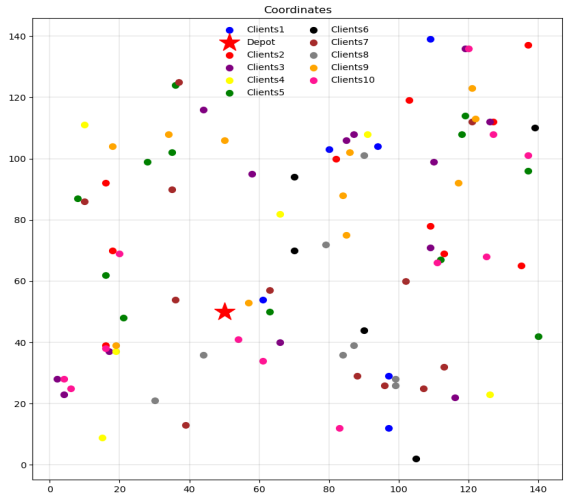

Figure 4.2: Deliveries randomly assigned to owners.

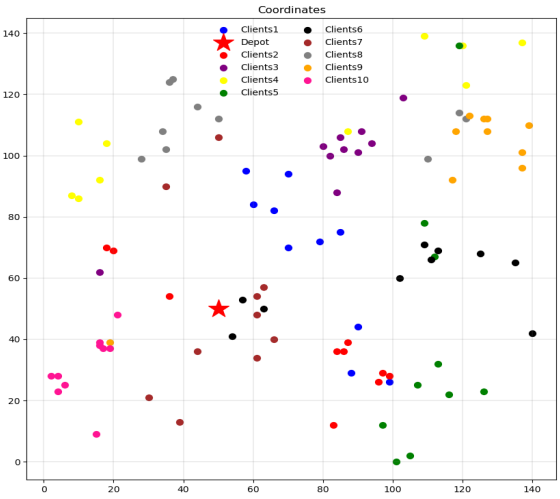

Figure 4.3: Deliveries assigned to owners in a clustered manner.

From the figures, it can be seen that when orders are assigned randomly to owners, any two companies could benefit from cooperation as deliveries are spread randomly across the entire area. In contrast, when deliveries are assigned to owners in a clustered manner, not all companies would benefit from cooperation. For instance, companies 1 and 10 are

unlikely to benefit, as their deliveries are located in significantly different areas on opposite sides of the depot.

The Gehring & Homberger instances include various different instances, where the deliveries are either randomly spread, spread in a clustered way or instances including deliveries spread in both random and clustered way. Figures 4.4, 4.5 and 4.6 show an example of these three scenarios.

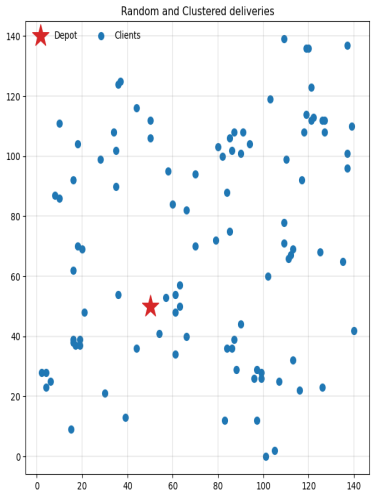

Figure 4.4: Random and clustered deliveries.

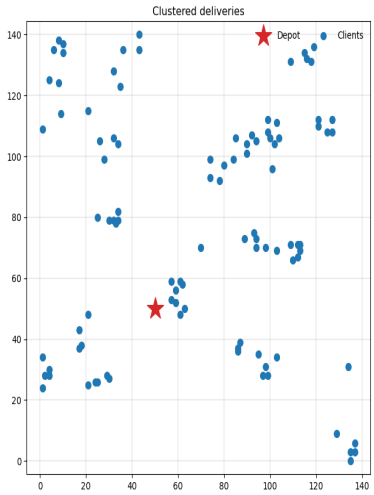

Figure 4.5: Clustered deliveries.

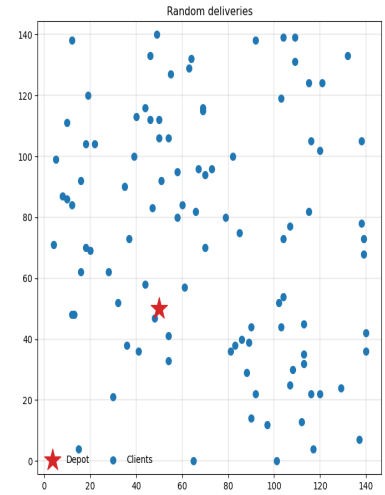

Figure 4.6: Random deliveries.

This study focuses on using various different instances to create a training set for the machine learning algorithm. To do so, the instances are split up into multiple instances. A total of 138 instances are created where each instance has different characteristics. Table 4.2 shows four instances out of the total 138, and table 4.1 summarises these scenarios. An overview over all the 138 instances used, can be found in the Appendix.

Characteristic	**Value:**
Number of Instances	138
Number of companies	[5, 10, 15, 20]
Capacity of the trucks	[10, 15, 20, 30, 60]
Total number of deliveries	[50, 100, 150, 200, 400, 500, 800]

Table 4.1: All potential values for instance characteristics.

Instance name	Deliveries	Capacity	Cost	Comp.	Depot
homberger_800/C1_8_1	100	15	100	10	[100, 100]
homberger_200/R1_2_1	200	20	0	10	[100, 100]
homberger_1000/C1_10_1	500	10	400	20	[400, 400]
homberger_1000/RC1_10_1	800	20	300	20	[300, 300]

Table 4.2: An example of four different instances.

4.1.2 Time comparison

For the algorithm created to be useful it needs to be able to solve the instances faster than it takes to solve every possible instance as is. This section will be dedicated to time comparison between the created algorithm and using PyVRP to solve every possible cooperation. The first subsection, 4.1.2.1 will be used to look at how much faster the slowest feature is compared to solving all possible cooperations. For this part, the time for every possible cooperation of two parties will only be considered, as the slowest feature is only used for the first step in the iterative method. The second subsection 4.1.2.2, will then be used to look at the total time comparison between solving all possible cooperations for up to four collaborative parties exactly, compared to the time it takes to create the features within the iterative method.

4.1.2.1 Solving all VRP's compared to VRP's between clusters

The idea behind clustering the delivery points and then solving the vehicle routing problem between the cluster centroids is to have fewer deliveries to visit, allowing for the heuristic algorithm to scale better, where it can handle larger instances. Figures 4.7 and 4.8 show the difference in runtime for running the algorithm for any possible cooperation of two partners. The blue line in the plots shows the exact time, where no alternations have been made to the deliveries and a vehicle routing problem is solved for all those deliveries. The orange line shows the case where all deliveries within a cluster have been combined into one large delivery in the centroid of the cluster. This delivery is equal to the size of all the other deliveries within the cluster combined. Similarly, the green line shows the instance where all deliveries have been moved to the centroid, but not combined. This allows for split deliveries where two or more trucks can handle the orders of one cluster. The lines in the figures show the average time of 10 independent runs and the shadow is the standard deviation of those runs. All the runs are made on the

homberger_1000_customer_instances/RC1_10_1.TXT file with the first delivery being the first one of the instance and the last equal to the total number of deliveries.

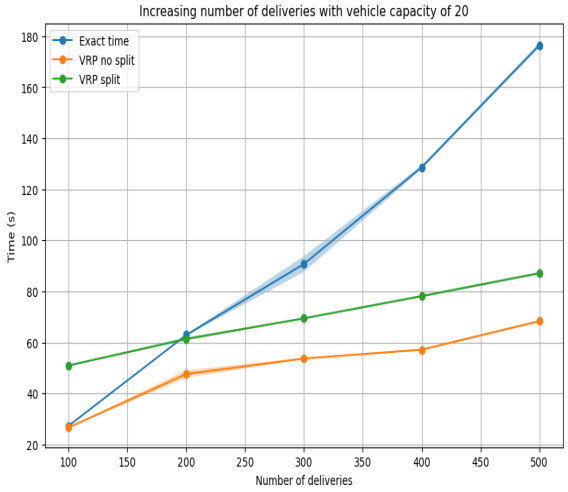

Figure 4.7: Increasing number of deliveries for ten companies, all with vehicles of capacity 20.

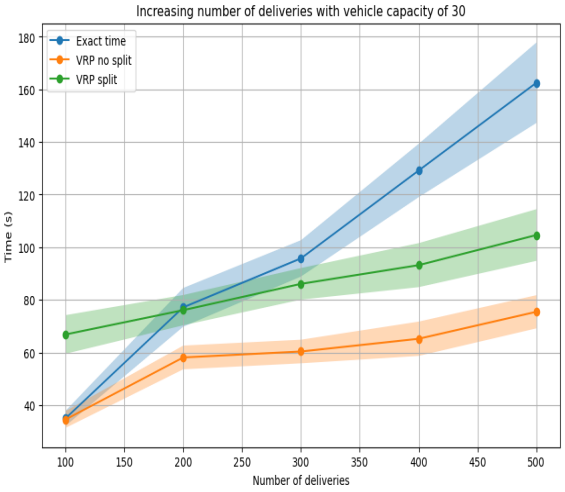

Figure 4.8: Increasing number of deliveries for ten companies, all with vehicles of capacity 30.

The figures show that clustering the deliveries, before solving the vehicle routing problem, results in less computational time compared to solving all possible collaborations to an exact solution. It may be noticed that when there are few deliveries the difference between the exact solution and the *VRP between clusters* is not very high, where allowing for split deliveries even results in more computational time compared to the exact solution. However, with increase in total number of deliveries the difference becomes larger, emphasizing the benefits of the clustering before solving the vehicle routing problem.

4.1.2.2 Creating all features compared to solving all VRP's

Subsection 4.1.2.1 shows that utilizing clustering to make the vehicle routing instance smaller, speeds up the process of getting an estimated route. This also shows that allowing for split deliveries requires more computational time compared to not allowing for split deliveries. Therefore, not allowing for split deliveries and only visiting the cluster centroids allows for larger instances being able to be estimated in less time. It should also be noted that the time comparison in subsection 4.1.2.1 only shows the time it takes to solve the VRP between cluster centroids compared to the VRP between all deliveries, only for partnerships of size two. Even though this feature is quicker to compute than solving

the vehicle routing problem for all deliveries, it is still a slow feature. Table 4.3 shows how many possible combinations there are for various number of companies and different partnership sizes. Figure 4.9 shows the values from the table in a line plot.

Count of companies	Two partners	Three partners	Four partners
5	10	10	5
10	45	120	210
15	105	455	1365
20	190	1140	4845
25	300	2300	12650
30	435	4060	27405
35	595	6545	52360
40	780	9880	91390

Table 4.3: Increase in possible partnerships with increasing number of companies.

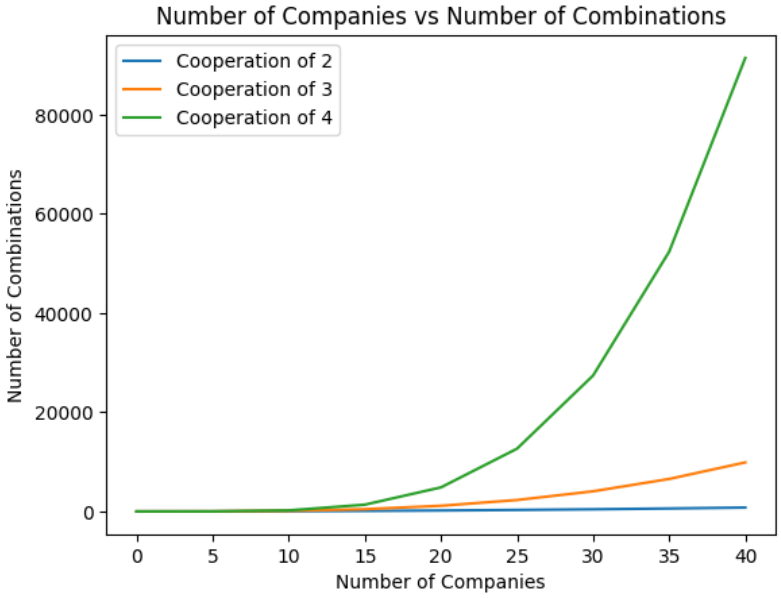

Figure 4.9: Increase in possible partnerships with increasing number of companies.

From this table it can be seen that for increase in number of companies to be considered, the number of possible partnerships grows very quickly. However, the larger the partnerships are, the faster the growth. This can be seen from looking at the instance with 40 companies. By only allowing partnership of two companies, there are 780 possible combinations, however, by allowing for partnerships of 4 companies this number grows to

91.390. For this reason, it is important to speed up the calculations when the partnerships become larger. Therefore, the slowest features will only be computed for partnerships of two companies, this includes all features utilizing the vehicle routing problems between cluster centroids.

This means that the slowest features, utilizing the vehicle routing problems between cluster centroids, is only computed for the first step in the iterative method described in 3.5. When the prediction of potential gains has been made for all possible partnerships of two companies, the predictions will be used to compute new features that take a lot less time to compute. Figures 4.10 show the difference between the time it takes to solve all possible combination of cooperative partners for up to four companies, compared to the time it takes to create all the features needed to get those same predictions. The time comparison was made on instance *homberger_1000_customer_instances/C1_10_1.TXT* where each company has 20 deliveries and the capacity of the vehicles is fixed to 30. The lines in the plot show the average runtime over five independent runs.

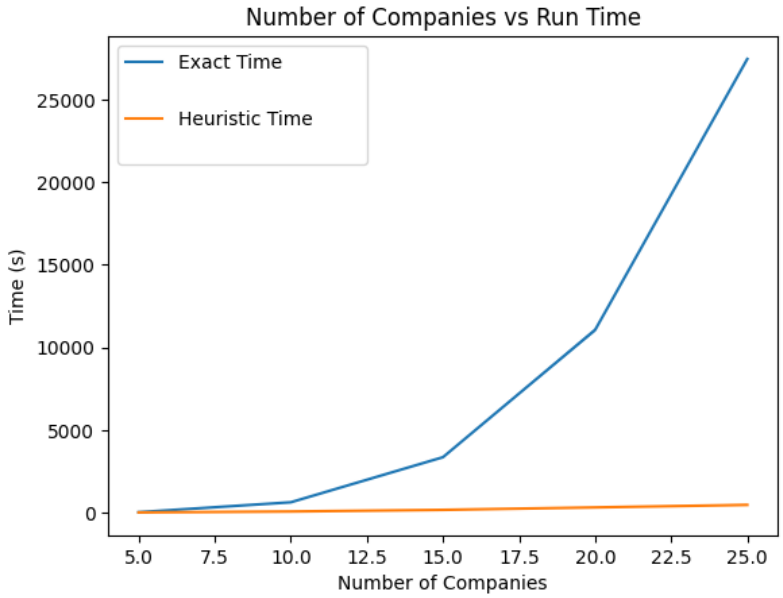

Figure 4.10: Comparison of run time with increasing number of companies.

Since the algorithm used to identify the potential collaborative partners only utilizes the *VRP between clusters* for cooperation between two companies, all other features are computed very quickly in comparison. Consequently, the runtime of the algorithm follows the growth trend for cooperation between two companies, this can be seen in figure 4.11 where the orange line, the runtime of the algorithm, follows the trend of the dashed red

line, the number of combinations for cooperation of two companies. Similarly, the exact computational time, the blue line in figure 4.11 follows the yellow dashed line, which represents the total number of collaborations for combinations of up to four companies.

However, it should be noted that these lines only illustrate the growth trends of runtimes in comparison to the growth trends of the number of possible combinations. Specifically, figure 4.11 shows that the runtime for the exact solution increases similarly to the trend of the sum of all collaborations, while the growth of the created algorithm aligns with the number of combinations involving two companies. Therefore, this graph is only used to give an indication of how quickly the computational time increases.

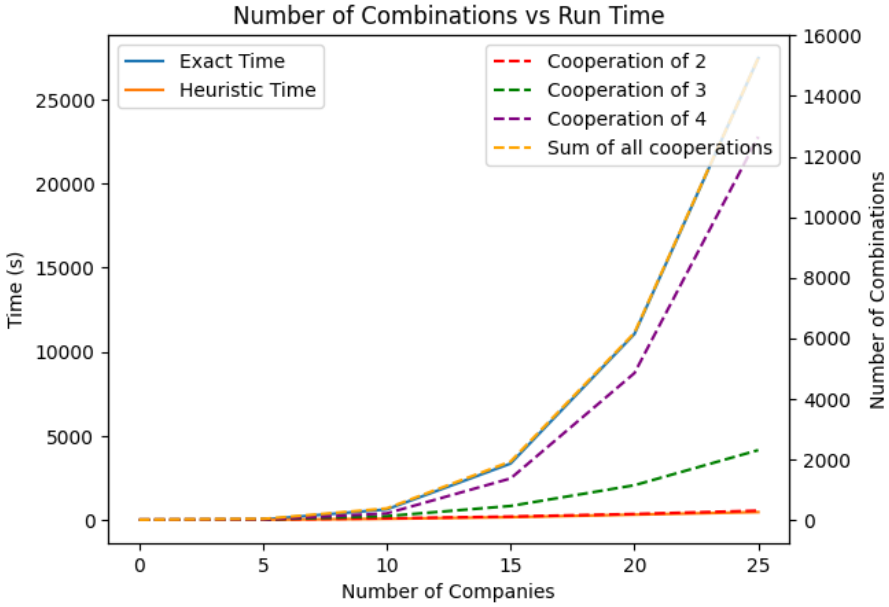

Figure 4.11: Comparison of run time and the number of combinations with increasing number of companies.

Both figures 4.9 and 4.10 only show the computation time for cooperation of maximum four companies. As the algorithm is iterative, it can handle larger collaborations. Looking at the trends, it can be expected that the computational time for the heuristic will follow close to the growth trend for cooperation between two companies, while the exact solution will follow the trend of the sum of all collaborations.

4.1.3 Performance metrics

To assess the performance of the machine learning algorithms, multiple metrics will be employed. The first two, commonly used in regression models, are the root mean square

error (RMSE) and the R^2 score. These metrics are unranked, meaning they measure how closely the algorithms' predictions match the true values. However, since this study also focuses on the algorithms' ability to distinguish between promising and unpromising collaborations, ranked performance metrics will also be considered. These include Precision@K, Precision@K%, and a newly introduced metric, called Directional Trend Accuracy (DTA).

Coefficient of determination ($\mathbf{R}^2$) R^2 measures the proportion of the variance in the dependent variable that is predictable from the independent variables. It normally ranges from 0 to 1, with higher values indicating a better fit. However, it should be noted that R^2 can be negative. This occurs when the predictions fit the data worse than a horizontal line representing the mean of the dependent variable. This means that the model explains less of the variance in the data than a simple average would. R^2 is obtained as follows:

$$R^2 = 1 - \frac{SS_{res}}{SS_{tot}}$$

SS_{res} is the residual sum of squares, the sum of the squares of the differences between the observed and predicted values. SS_{tot} is the total sum of squares, the sum of the squares of the differences between the observed values and the mean of the observed values.

Root mean squeare error (RMSE) is a commonly used performance metric, where it measures the difference between the true and the predicted values. In this case the difference between the predicted gains and the true gains. RMSE measures the average magnitude of the errors between the predicted and actual values. It provides a sense of how well the model's predictions match the actual data, with lower values indicating a better fit. RMSE is obtained by calculating:

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{n}} = \sqrt{\frac{SS_{res}}{n}}$$

Where n is the total number of observations. Similarly, y_i is the true value of observation i and $\hat{y}_i$ is the predicted value of observation i.

It should be noted that both RMSE and R^2 are unranked performance metricises. Therefore these two metrics are more representative of how close the models manages to predict the gains to the actual gains, than they are at representing the ranking of the predicted values. If the RMSE is high, it indicates that the residual sum of squares, SS_{res}, is high, meaning the model's predictions are not very close to the actual values. However, a high

SS_{res} means that the ratio $\frac{SS_{res}}{SS_{tot}}$ is high, which in turn makes $1 - \frac{SS_{res}}{SS_{tot}}$ lower. Therefore, when RMSE is higher, it results in a lower R^2. This is because usually a lower RMSE indicates a better fit, leading to a higher R^2. However, this can be misleading in our case as we are not only interested in the value itself, but the ranking of the values within their own instance. We want the predicted gains to be representative of the actual gains, such that negative gains are actually negative and positive are positive. However, if the ranking within the predictions are spot on, we also consider that a good prediction by the model.

Precision@K The previously mentioned performance metrics are unranked, focusing on the error of all predictions for all possible collaborations independently. However, this study primarily focuses on the ranking of the highest predicted collaborations and how close the predicted ranking is to the correct ranking. Precision at K (Precision@K) is a ranking metric commonly used to evaluate the performance of information retrieval and recommendation systems. It measures the proportion of relevant items in the top K results returned by a model. Precision@K is defined as the number of relevant items among the top K items divided by K.

$$Precision@K = \frac{\text{Number of relevant items in top } K}{K}$$

Where an item is relevant if the item is considered a correct or desired outcome.

- Top 10 in 10: In this case, $K = 10$ and the relevance list includes all ranks from 1 to 10, i.e., [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]. This metric evaluates whether the 10 highest predicted gains align with the 10 highest actual gains. For instance, if 6 out of the top 10 actual gains are predicted among the top 10 highest gains, the score would be 0.6.

- Top 10 in 15: In this case, $K = 10$ and the relevance list are all ranks from 1 to 15. This metric evaluates whether the top 10 actual gains are predicted among the top 15 predicted gains.

- Top 10 in 20: In this case, $K = 10$ and the relevance list includes ranks from 1 to 20. This metric checks if the top 10 actual gains are among the top 20 predicted gains.

- Top 20 in 20: Here $K = 20$ and the relevance list are all ranks from 1 to 20. This metric looks if the top 20 real gains are predicted in the top 20 predicted gains.

Precision@$K\%$ Since the instances vary in the number of companies and, consequently, the number of possible combinations of two, another metric is introduced. This metric evaluates the top 10% of the rankings. For example, if there are 100 possible combinations, the top 10 will be considered, whereas if there are 300 combinations, the top 30 will be evaluated.

- Top 10% in the highest 10%: In this case, $K = 10\%$, and the relevance list includes all ranks from 1 to k, where k represents the total number of values corresponding to the top 10% of all combinations.
- Top 10% in the highest 15%: In this case, $K = 10\%$, and the relevance list includes all ranks from 1 to k, where k represents the total number of values corresponding to the top 15% of all combinations.
- Top 10% in the highest 20%: In this case, $K = 10\%$, and the relevance list includes all ranks from 1 to k, where k represents the total number of values corresponding to the top 20% of all combinations.

Directional Trend Accuracy (DTA) The previous metrics focus solely on assessing whether the top K or top $K\%$ predictions are correct. However, it is also important to evaluate how well the algorithm ranks the entire set of collaborations. This means, that for the most accurate predictions, an ideal outcome would be a perfect ranking, where the predicted ranking exactly matches the true ranking. To address this, a new metric is introduced to provide insight into the algorithm's ranking performance. This metric, inspired by the Mean Directional Accuracy (MDA), a statistical measure often used in time series analysis, compares the predicted values to the true values. The MDA compares each value to its previous value to see if the values is increasing or decreasing. If the direction of change is same for the predicted values and the real values, it is considered to be a correct direction. To address this issue, a new metric called Directional Trend Accuracy (DTA) is introduced. The DTA penalizes deviations more severely based on how far the predicted ranking is from the true ranking, providing a more detailed evaluation of ranking accuracy.

Given prediction $\hat{y} = [\hat{y_1}, \hat{y_2}, ..., \hat{y_n}]$ and the true values $y = [y_1, y_2, ..., y_n]$, with n being the total number of collaborations in the instance, two counters are initialized:

- Correct directions = 0

- Total comparisons = 0

A function is then created that loops through each pair of indices i and j: (i, j), where $i < j$. For each pair the direction of change is compared for both the true values and the predicted values.

$$\text{TrueDirection}_{ij} = \begin{cases} +1 & \text{if } y_i < y_j, \\ -1 & \text{if } y_i > y_j \\ 0 & \text{if } y_i = y_j \end{cases}$$

$$\text{PredictedDirection}_{ij} = \begin{cases} +1 & \text{if } \hat{y}_i < \hat{y}_j, \\ -1 & \text{if } \hat{y}_i > \hat{y}_j \\ 0 & \text{if } \hat{y}_i = \hat{y}_j \end{cases}$$

If both $\text{TrueDirection}_{ij}$ and $\text{PredictedDirection}_{ij}$ have the same sign, the *correct directions* are incremented. For every comparison the *total comparison* is incremented. Then the DTA is computed.

$$DTA = \frac{\text{Correct directions}}{\text{Total comparisons}}$$

Since each value is compared with all subsequent values, this metric imposes a greater penalty the further the ranking is from the true ranking. It is important to note that if the ranking is entirely random, this metric will converge to 0.5. Thus, a score higher than 0.5 indicates that the algorithm performs better than random guessing, which can be used as a baseline for performance. Similarly, a perfect ranking yields a DTA score of 1. Ideally, the metric should range between 0.5 and 1, with a preference for values closer to 1.

4.1.4 Performance of the benchmark instances

This section presents the performance of the iterative method across the three machine learning models discussed in section 3.3. Tables 4.4 and 4.5 illustrate performance metrics for the unranked evaluations. Specifically, table 4.4 displays the results for the untuned models, while table 4.5, presents the results after tuning the models with the hyperparameters detailed in section 3.3.

RMSE and R^2 The tables present the average score and standard deviation calculated across all 138 benchmark instances. They also reveal that tuning the models enhances performance across all steps of the iterative method for every model used. The most

significant improvement is observed in the Decision Tree model, whereas the performance gains for the other two algorithms are relatively modest.

Examining the R^2 values in the tables reveals that the highest score for cooperation between two companies is achieved with both the tuned Random Forest and the tuned XGBoost models, reaching a value of 0.72. This suggests a relatively good fit on average. However, the RMSE values may be affecting the R^2 scores, resulting in lower values. It is also noteworthy that the R^2 metric decreases significantly in the iterative method for predicting gains for cooperation among three companies, dropping to around 0.4 for both the tuned Random Forest and tuned XGBoost models. These high RMSE values and low R^2 scores in the iterative steps indicate challenges in predicting exact gains. Nonetheless, it is important to consider that in the benchmark instances, the gains for cooperation of two companies range from a maximum of 1502 to a minimum of -244. For cooperation of three companies, the range is from 2956 to -388, and for cooperation of four companies, from 4190 to -486. Thus, an average RMSE around 100 is relatively reasonable given the wide range of values.

The standard deviation of the R^2 values should be interpreted with caution, as some cases show values exceeding 1.0. Since R^2 cannot exceed 1.0, this suggests that the distribution is right-skewed, with a greater concentration of instances near 1.0.

Untuned		**RMSE**		**R^2**	
		Average	**Std**	**Average**	**Std**
Decision Tree	Cooperation of two	104.60	51.70	0.52	0.40
	Cooperation of three	192.26	104.48	-0.14	1.15
	Cooperation of four	262.17	202.21	-1.69	6.06
Random Forest	Cooperation of two	82.82	45.76	0.70	0.28
	Cooperation of three	144.901	80.41	0.35	0.84
	Cooperation of four	202.86	141.92	-0.14	1.94
XGBoost	Cooperation of two	79.55	48.43	0.71	0.28
	Cooperation of three	139.74	76.09	0.38	0.75
	Cooperation of four	188.37	141.68	-0.10	2.19

Table 4.4: How many of the best cooperation pairs are predicted as the best ones in XGBoost iterative algorithm. Average of 138 instances. Before tuning the models.

Precision@K The previous tables showed performance of the benchmark instances in terms of unranked metrics. While the results were reasonably good, specifically in the

Tuned		RMSE		R^2	
		Average	Std	Average	Std
Decision Tree	Cooperation of two	93.04	46.88	0.62	0.36
	Cooperation of three	173.95	92.68	0.07	1.12
	Cooperation of four	206.42	140.54	-0.25	1.86
Random Forest	Cooperation of two	80.62	43.98	0.72	0.27
	Cooperation of three	137.42	76.34	0.42	0.70
	Cooperation of four	191.76	148.36	-0.09	1.62
XGBoost	Cooperation of two	78.55	47.36	0.72	0.37
	Cooperation of three	138.37	86.70	0.40	0.37
	Cooperation of four	179.70	108.94	-0.02	1.75

Table 4.5: How many of the best cooperation pairs are predicted as the best ones in XGBoost iterative algorithm. Average of 138 instances.

first step of the iterative method, focusing solely on unranked metrics might be somewhat misleading. Therefore it is important to evaluate how effectively the algorithm ranks the highest values correctly, rather than focusing only on the accuracy of the exact predicted gains. Figure 4.12 presents the performance of the untuned models for various K values and for two companies' cooperation. The bars show the average performance of all 138 benchmark instances, with the black line showing the standard deviation. Notably, the untuned models often show average precision values above 0.6, suggesting that the models correctly identify at least 60% of the most promising collaborations. In fact, XGBoost and Random Forest models achieve between 80% and 90% accuracy in these predictions. These results are quite promising, even for untuned models, as indicated by the red dotted line representing the highest possible precision.

For the next step of the iterative method, which involves collaborations of three companies, seen in figure 4.13, the precision values decrease slightly. However the average precision remains above 50% in most cases for both the Random Forest model and the XGBoost, and it even exceeding 70% in some cases, indicating promising performance with the untuned models. Similarly, for collaborations of four companies, illustrated in figure 4.14, the precision is generally around 50-60%, with some instances reaching close to 70%.

The untuned models show promising results. However, figures 4.15, 4.16 and 4.17 illustrate the precision@K for the tuned models. These figures reveal a significant improvement in precision with model tuning. Specifically, figure 4.15 shows that precision values are gen-

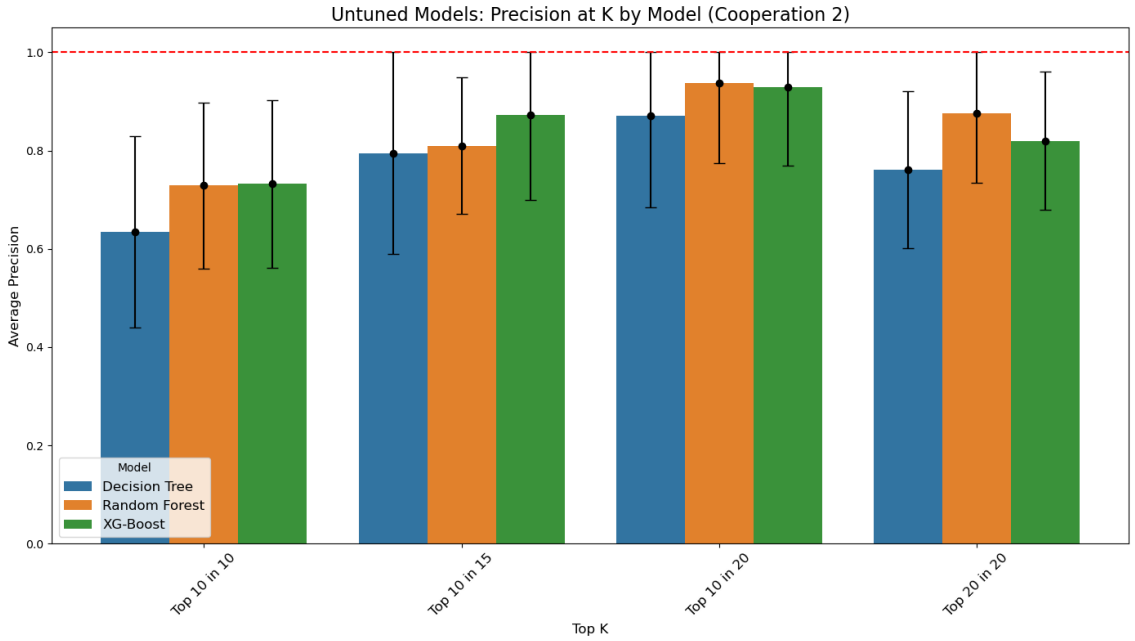

Figure 4.12: Precision at K for cooperation of two: untuned models.

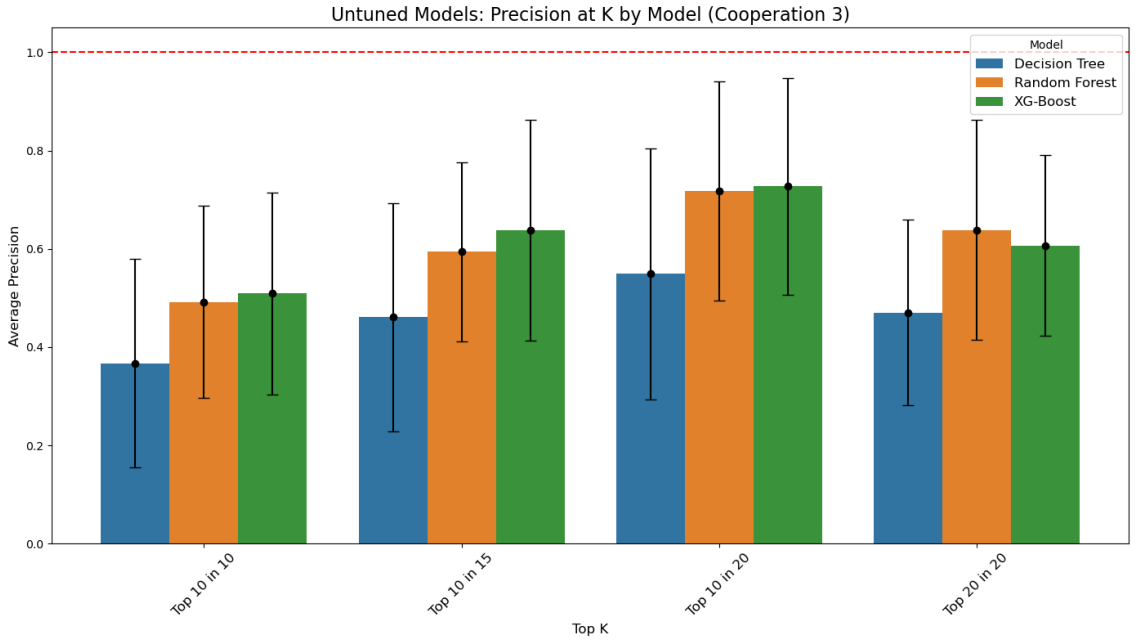

Figure 4.13: Precision at K for cooperation of three: untuned models.

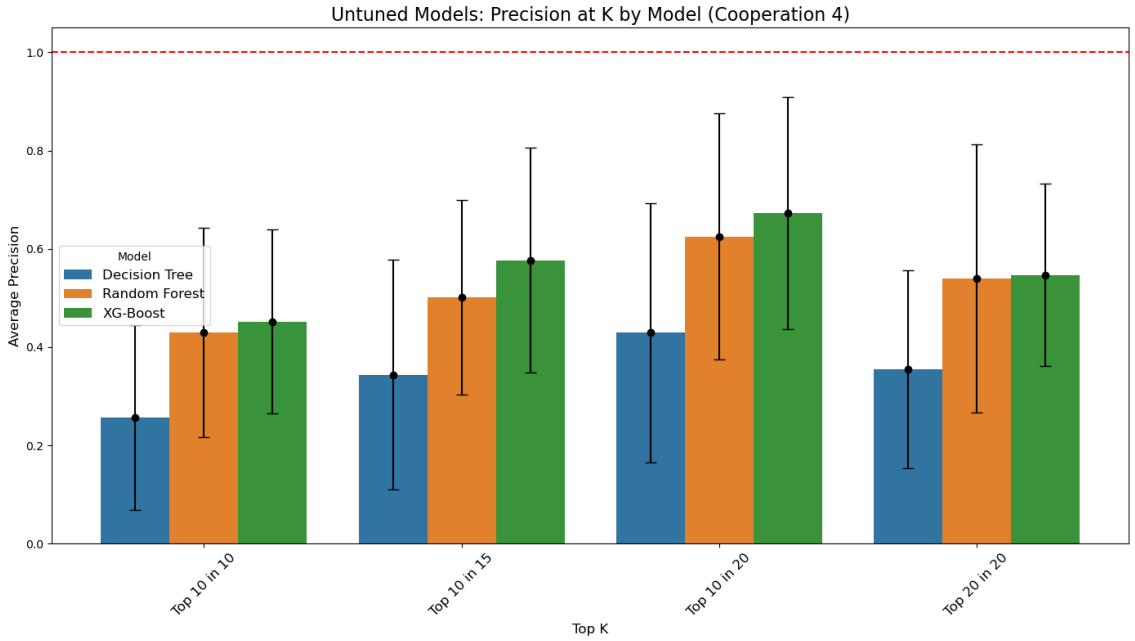

Figure 4.14: Precision at K for cooperation of four: untuned models.

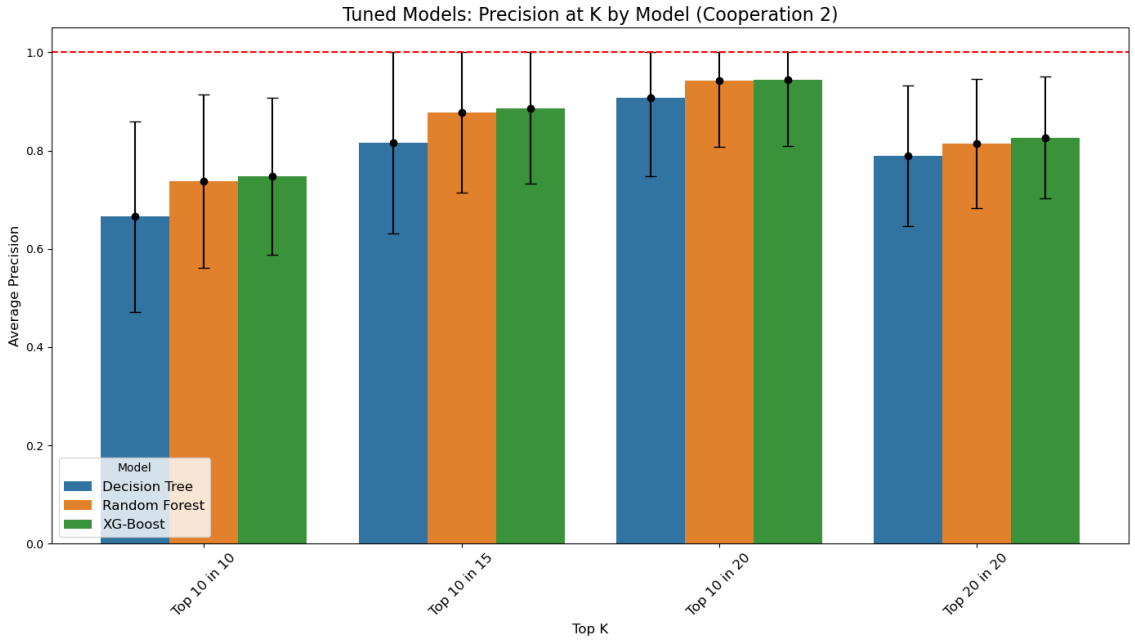

Figure 4.15: Precision at K for cooperation of two: tuned models.

erally above 65% across all models, with Random Forest and XGBoost achieving precision between 74% and 94%. This indicates that, in most cases, the tuned models manage to effectively identify all best collaborations, yielding very strong results.

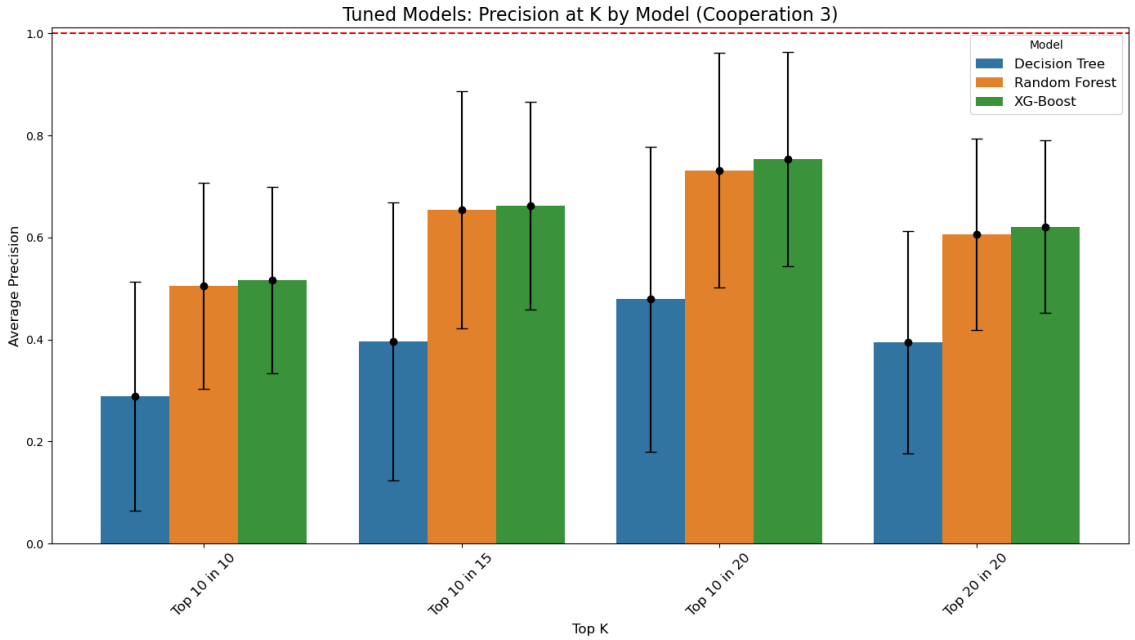

Figure 4.16: Precision at K for cooperation of three: tuned models.

Figures 4.16 and 4.17 show precision@K for the subsequent steps of the iterative method, where the average precision for both XGBoost and Random Forest ranges from 50% to 75%. In contrast, the Decision Tree algorithm exhibits significantly worse results, suggesting that it may be too simplistic to effectively capture patterns within the data and provide accurate predictions. Overall, the tuned models demonstrate strong performance in accurately identifying the most promising collaborations.

Precision@K% As previously mentioned, the number of companies in the instances, and consequently, the number of possible two-company combinations, varies. Therefore, it is also useful to evaluate the algorithms based on the highest 10% of predictions rather than just the top 10 instances. Figures 4.18, 4.19 and 4.20 illustrate performance across all four steps of the iterative method.

Notably, for collaborations of two companies, XGBoost excels, achieving an average precision of 90% when predicting the top 10% within the highest 20% of collaborations. It also reaches an average accuracy of 69% for predicting the top 10% within the highest 10%.

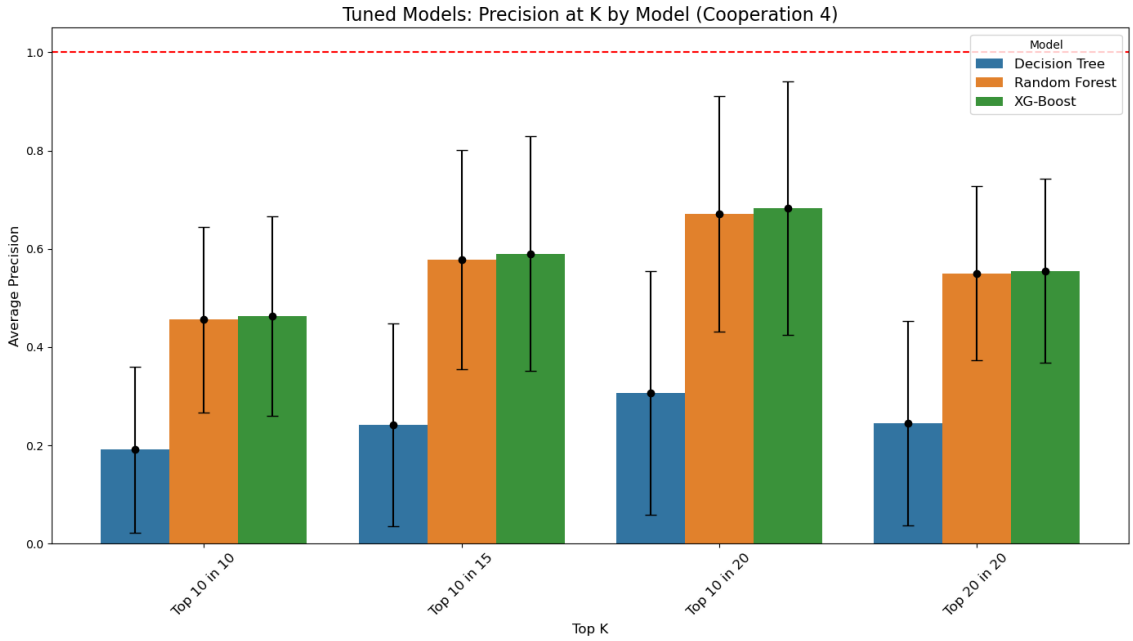

Figure 4.17: Precision at K for cooperation of four: tuned models.

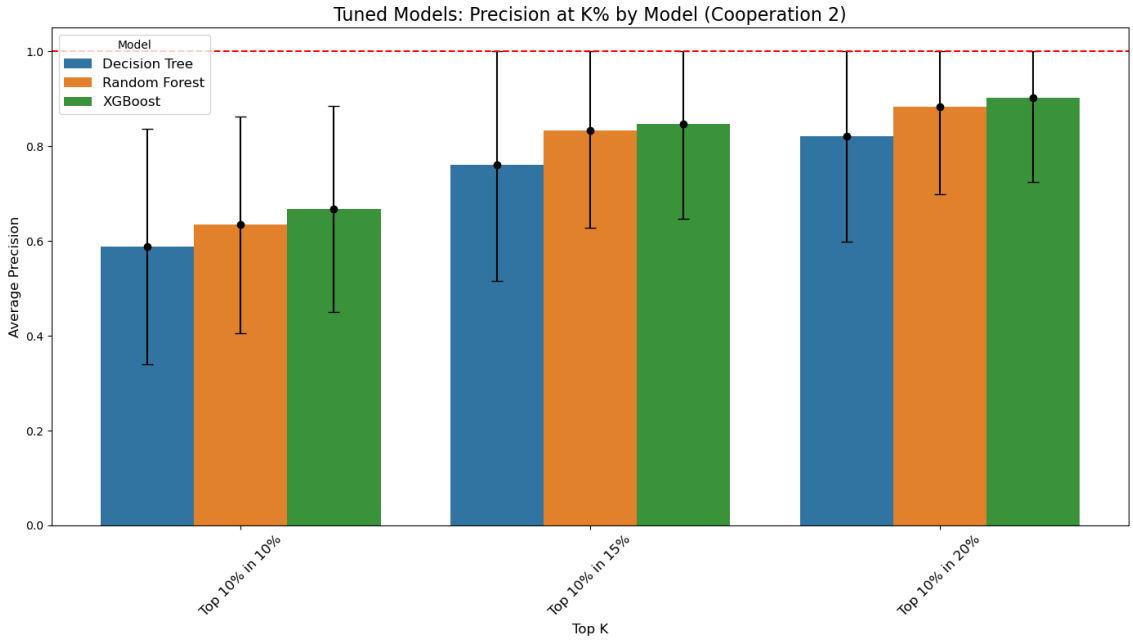

Figure 4.18: Precision at $K\%$ for cooperation of two: tuned models.

Random Forest performs similarly but with slightly lower precision. In contrast, the Decision Tree algorithm consistently underperforms. For both Random Forest and XGBoost, the next steps of the iterative method, including cooperation of three and four companies, seem to decrease less in performance compared to the performance when looking at precision@K. This could be because, as collaboration sizes increase, the number of possible combinations rises significantly, as shown in table 4.3. For instance, with 30 companies, there are 4.060 possible three company partnerships and approximately 27.000 possible four company partnerships. Focusing on only the top 10 or 15 values for collaborations involving three or four companies might be insufficient, as these top values are all highly promising, potentially making them comparable to the next set of values. However, by considering the highest 10%, which includes the top 400 values for three company collaborations, the distinction between promising and less promising collaborations becomes clearer. This suggests that while the algorithm may effectively identify overall patterns in promising collaborations, it may struggle to differentiate between collaborations with very similar gains.

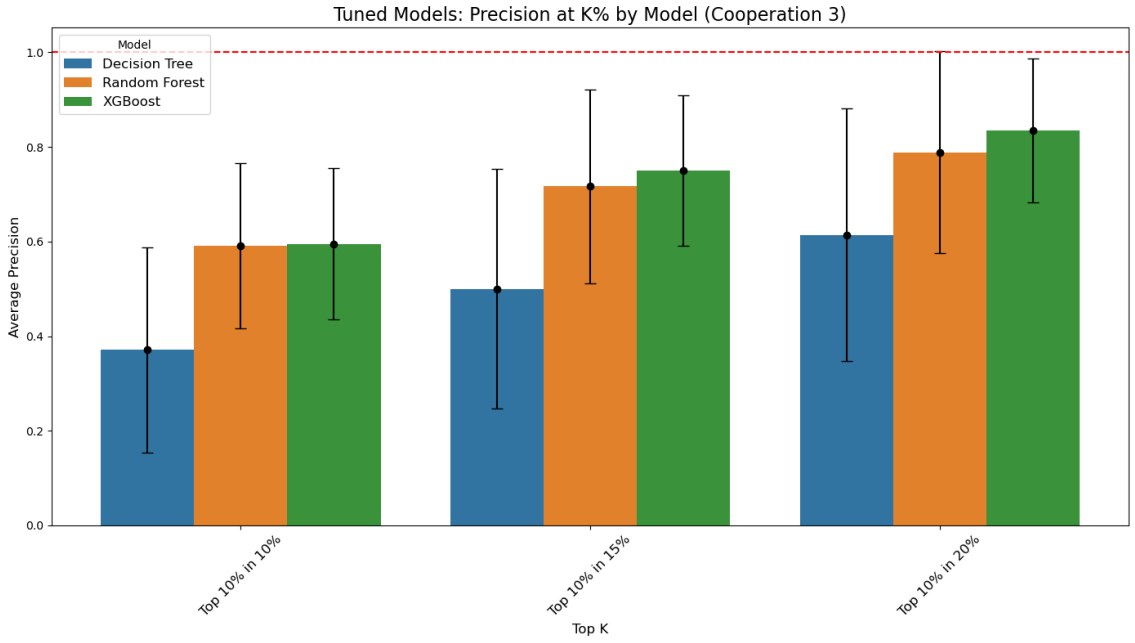

Figure 4.19: Precision at $K\%$ for cooperation of three: tuned models.

The detailed values for all the precision@K and precision@K% figures are provided in the Appendix.

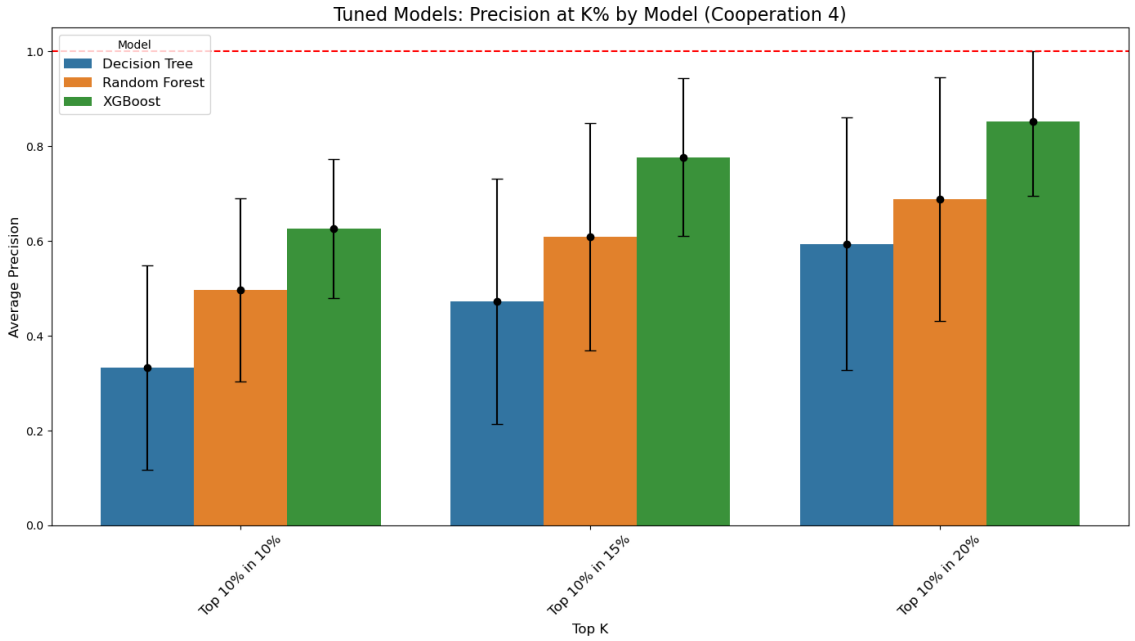

Figure 4.20: Precision at $K\%$ for cooperation of four: tuned models.

Directional Tend Accuracy (DTA) Table 4.6 presents the DTA scores for each step of the iterative method across all tuned machine learning models. It is particularly noteworthy to compare these scores with the R^2 values shown in table 4.5. When evaluating based solely on exact gains, the performance of the iterative method appears to decrease significantly with each iterative step, indicating that predictions increasingly become further from the true values. However, when assessed in terms of ranking accuracy, the performance decrease is less dramatic. The Decision Tree algorithm's performance for collaborations of three and four companies is close to random ranking (0.5), whereas Random Forest and XGBoost demonstrate way better ranking results. Nonetheless, all algorithms outperform random predictions in all iterative steps.

This indicates that the algorithm has some difficulty in accurately predicting the exact expected gains but performs significantly better when it comes to ranking the collaborations. This observation supports the notion that RMSE and R^2 metrics might be somewhat misleading. The key factor is not whether the algorithm predicts the exact gains accurately, but rather whether it can rank the collaborations correctly. The algorithm appears to do well in ranking, even if its precision in predicting exact values is less reliable.

Figure 4.21 visualises the DTA scores for each step of the iterative method across all

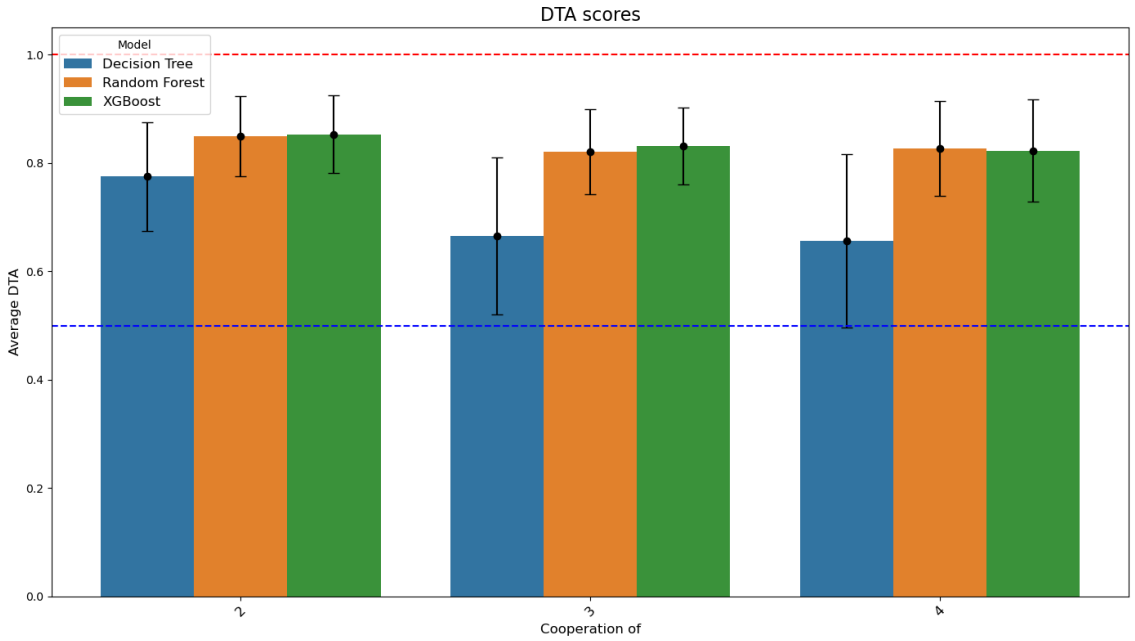

Figure 4.21: DTA for all models and four steps of the iterative method.

models. The red line denotes the highest possible DTA score, while the blue line represents a score corresponding to a completely random ranking. As previously observed, it can be seen from the figure that the DTA scores remain relatively consistent across each step of the iterative method, indicating that the algorithm manages to capture the ranking of the collaborations quite well in each step. Both the Random Forest algorithm and the XGBoost achieve DTA values over 0.8 in all steps, suggesting significantly better ranking performance compared to random.

4.1.5 Feature importance

One interesting aspect of the chosen machine learning algorithms is their ability to find the importance of each feature in predicting the expected gains. This offers additional insight into which features effectively identify good collaborating partners and which ones fail to do so. This section will analyze the features for each step of the iterative method to evaluate their effectiveness. Additionally, it will include a discussion on the features, providing possible explanations for why some perform well while others do not.

All results in this section are obtained using the XGBoost algorithm. While the importance scores are similar for the other two algorithms (Decision Tree and Random Forest),

		DTA: Average	DTA: Std
Decision Tree	Cooperation of two	0.775	0.100
	Cooperation of three	0.666	0.145
	Cooperation of four	0.656	0.160
Random Forest	Cooperation of two	0.850	0.074
	Cooperation of three	0.821	0.079
	Cooperation of four	0.827	0.087
XGBoost	Cooperation of two	0.853	0.072
	Cooperation of three	0.831	0.071
	Cooperation of four	0.823	0.094

Table 4.6: The average DTA and the standard deviation for all 138 instances.

the focus is on XGBoost as it achieved the highest accuracy among the three. However, feature importance for the other algorithms is available in the Appendix. The importance scores were determined using tuned models, which were trained on 80% of the benchmark instances mentioned in section 4.1.1 and tested on the remaining 20%.

4.1.5.1 Cooperation of two

As previously mentioned, for cooperation of two companies, all features discussed in section 3.2 are utilized, including those computing the vehicle routing problem between the cluster centroids to which the deliveries belong. Since these are the most time-consuming features to generate, they are not created for all possible collaborations of all sizes. It is quite interesting to observe which features provide the most insight into identifying promising collaborations.

Table 4.7 shows the importance of each feature in predicting gains for cooperation between two companies, using XGBoost. Some features, including *percentage of clusters needing more than a truck*, *number of clusters needing more than a truck*, *VRP between clusters cost per company*, and *collaboration size*, had an importance of zero and thus had no impact on the results. Consequently, they were excluded from the table.

It is interesting to see that the feature *VRP between clusters cost before and after cooperation with fixed cost* seems to the most important feature in predicting the expected gains. This is expected, as it directly reflects the routes upon which the gains are based. The second most important feature is *truck capacity*, indicating that the size of the trucks significantly influences the potential benefits of collaboration. This is logical because if two trucks from different companies are sent to the same region, a collaboration is only

Feature	Importance
VRP between clusters cost before and after cooperation with fixed cost	0.701376
Truck capacity	0.041625
VRP between clusters cost before and after cooperation	0.038607
Trucks saved	0.036934
Coalition cost	0.033798
Instance number	0.027135
Silhouette score	0.020693
Total cost before collaboration	0.015617
Total truck utilization	0.013921
Number of clusters	0.013439
Percentage route sharing: considering delivery count	0.012279
Average distance to depot	0.009026
VRP between clusters	0.008739
Trucks saved based on clusters	0.005736
DBSCANscore	0.005301
Total number of deliveries	0.004582
KmeansScoreBinary	0.004557
Percentage route sharing	0.004088
KmeansScore	0.002549

Table 4.7: Feature Importance when predicting gains for cooperation of two using XGBoost.

beneficial if the combined deliveries from both companies can fit into a single truck. Otherwise, the collaboration essentially results in the same number of trucks as before, yielding minimal benefits. This is further supported by the importance of the *trucks saved* feature, which ranks just after the truck capacity in significance.

4.1.5.2 Cooperation of three

For cooperation of three not all features from section 3.2 are utilized, this is mainly due to the fact that as shown in table 4.3, the number of possible partnerships for cooperation of three companies is way larger than when only two companies are considered. For that reason the slowest features, the features computing the vehicle routing problem between cluster centroids, are neglected. However, new features are added to the dataframe. These are features that are computed from the predicted gains from cooperation of two companies. In predicting the expected gains from cooperation of three companies, table 4.8 shows the

importance of the features. Again, features *percentage of clusters needing more than a truck* and *collaboration size* had importance of zero, leading to them being excluded from the table.

Feature	Importance
Average of all	0.290797
Average of highest two subcollaborations	0.284893
Truck capacity	0.167352
Total truck utilization	0.032212
Silhouette score	0.025555
Trucks saved	0.023085
Number of clusters	0.021992
Average of highest two ranks	0.020903
Total cost before collaboration	0.020714
Percentage route sharing: considering delivery count	0.011614
Average distance to depot	0.011424
KmeansScore	0.010674
Instance number	0.009657
KmeansScoreBinary	0.008025
DBSCANscore	0.007738
Coalition cost	0.006978
Trucks saved based on clusters	0.006847
Subcollaboration 1	0.006175
Rank of subcollaboration 3	0.005400
Percentage route sharing	0.005242
Subcollaboration 3	0.004899
Total number of deliveries	0.004867
Rank of subcollaboration 1	0.002921
Highest subcollaboration	0.002762
Number of clusters needing more than a truck	0.002720
Subcollaboration 2	0.002653
Rank of subcollaboration 2	0.001901

Table 4.8: Feature importance for cooperation of three: XGBoost.

For cooperation involving three companies, identifying the most critical features is less straightforward compared to the previous step. In this case, the average of all predicted gains for each sub-collaboration within the overall collaboration, combined with the average

of the two highest sub-collaborations, appears to be key in determining the best cooperative partners. Additionally, features such as *truck capacity*, *total truck utilization*, and *trucks saved* also prove to be significant.

4.1.5.3 Cooperation of four

Just as with cooperation among three companies, not all features from section 3.2 are utilized. Similarly, new features were created based on the predicted gains from the previous step. Table 4.9 presents the feature importance for predicting the best collaboration among four companies.

Feature	Importance
Lowest subcollaboration	0.284734
Average of all	0.276859
Average of highest two subcollaboration	0.269438
Truck capacity	0.030168
Coalition cost	0.024309
Highest subcollaboration	0.021733
Average of highest three subcollaboration	0.019868
Rank of subcollaboration 1	0.017701
Subcollaboration 1	0.010050
Rank of subcollaboration 2	0.006814
Total cost before collaboration	0.004808
Average of highest two ranks	0.004397
Rank of subcollaboration 4	0.004376
Rank of subcollaboration 3	0.004144
Number of clusters	0.003958
Subcollaboration 4	0.003663
Instance number	0.003301
Average distance to depot	0.002848
subcollaboration 3	0.002557
Average of highest three ranks	0.002004
Total number of deliveries	0.001189
Subcollaboration 2	0.000662
Number of clusters needing more than a truck	0.000419

Table 4.9: Feature importance for cooperation of four: XGBoost.

When examining feature importance for cooperation involving four companies, the most

significant feature is somewhat unexpected. It appears that the *lowest subcollaboration* is the most crucial, suggesting that the potential success of a collaboration largely depends on the least favorable pair within the group. Along with this feature, *average of all* and *average of the highest two* are also key in identifying the best collaborations. These two features were also the most important when evaluating collaborations involving three companies. Notably, the *average of highest three subcollaborations* plays a minimal role in determining the most promising partnerships.

4.1.6 Assignment

The previous results demonstrate how effectively the algorithm predicts expected gains, both in terms of accuracy and in identifying the most promising collaborations. However, it might also be of interest to see how well the algorithm assigns companies to collaborations in a way that maximizes total gains from all collaborations. For a group of companies, the goal is to assign all companies to collaborations that maximize overall gains. This is achieved by applying the linear programming model discussed in section 3.6.1.1.

4.1.6.1 Performance metrics

Exact match ratio This metric measures the proportion of assignments that are exactly the same in both the real and predicted cases. It is a straightforward way to see how often the two assignment sets agree.

$$\text{Exact match ratio} = \frac{\text{Number of exact matching pairs}}{\text{Total number of pairs}}$$

Total gains difference This metric compares the true gains of the predicted assignments to the true gains of the optimal assignments. It measures how close the predicted total gains are to the actual total gains. A difference of zero indicates that the true gains of the predicted assignments match those of the optimal assignments. The closer the value is to zero, the better the algorithm's performance. This metric shows that even if the assignments are not exactly the same as the optimal ones, a value close to zero indicates that the predictions are still highly accurate, potentially being the second best or close to one of the best possible assignments.

$$\text{Total gains difference} = \text{True gains predicted assignment} - \text{True gains optimal assignments}$$

4.1.6.2 Results

This section presents the overall results from the assignment task, including both the exact match ratio and the total gains difference. For a clearer view of all the graphs and tables for each algorithm, they can be found in the Appendix.

Table 4.10 presents the average and standard deviation of the exact match ratio for all algorithms and collaboration sizes. Each algorithm achieves approximately 50% accuracy on average for assignments when identifying combinations of collaborations of size two or smaller, with the Decision Tree algorithm performing the best. However, when assignment tasks involve larger partnerships, such as those including up to three or four companies, the Random Forest and XGBoost algorithms tend to perform better on average, achieving about 25% accuracy. Figures 4.22, 4.23 and 4.24 illustrate the exact match ratio for all 138 benchmark instances. These figures indicate that for collaborations involving two or fewer companies, the algorithm identifies approximately one-third of the optimal assignments. However, as cooperation expands to three or four companies, as shown in figures 4.23 and 4.24 the exact match ratio drops significantly. This decline could be attributed to the increase in possible combinations as larger collaborations are allowed, resulting in a higher likelihood that the predicted combinations of collaborations will differ from the true optimal ones.

	Decision Tree		**Random Forest**		**XGBoost**	
	Average	Std	Average	Std	Average	Std
Cooperation of two	0.597	0.339	0.594	0.342	0.527	0.334
Cooperation of three	0.192	0.287	0.253	0.333	0.253	0.333
Cooperation of four	0.124	0.251	0.242	0.345	0.111	0.242

Table 4.10: Exact match ratio: Decision Tree, Random Forest and XGBoost.

Although the algorithm may not identify the optimal assignments for all collaborations, it may still be capable of finding high-quality solutions. Table 4.11 presents the difference between the true gains of the predicted assignment and the true gains of the actual best assignment. These values show how close the algorithm manages to find good combination of collaborations. Observing the min and max values, It can be observed that the average absolute error is relatively low when two or fewer companies cooperate. However, interestingly, the error values increase significantly when the algorithm permits more companies to collaborate. This suggests that in most cases involving collaborations of two or fewer companies, the algorithm successfully identifies the optimal or nearly optimal combinations

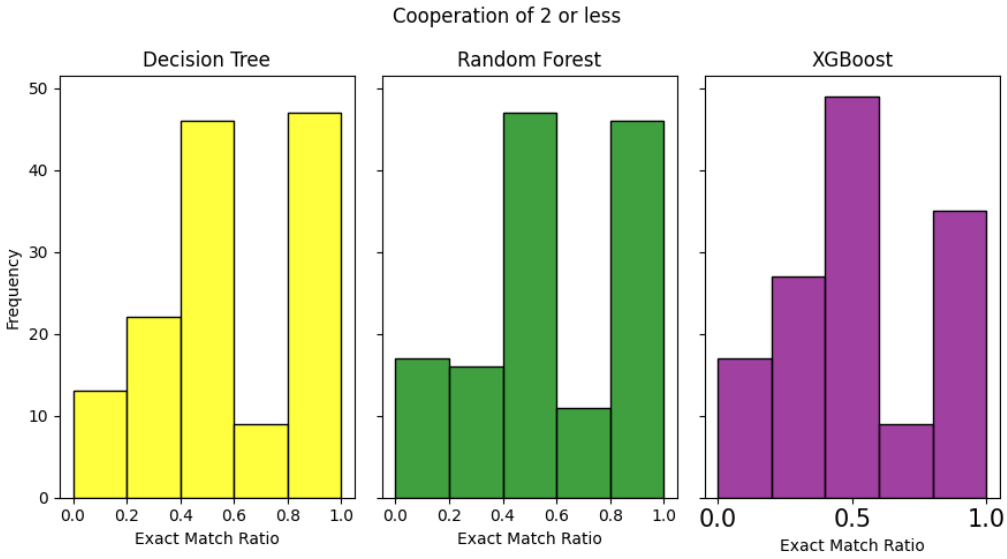

Figure 4.22: Exact match ratio for all models. Cooperation of 2.

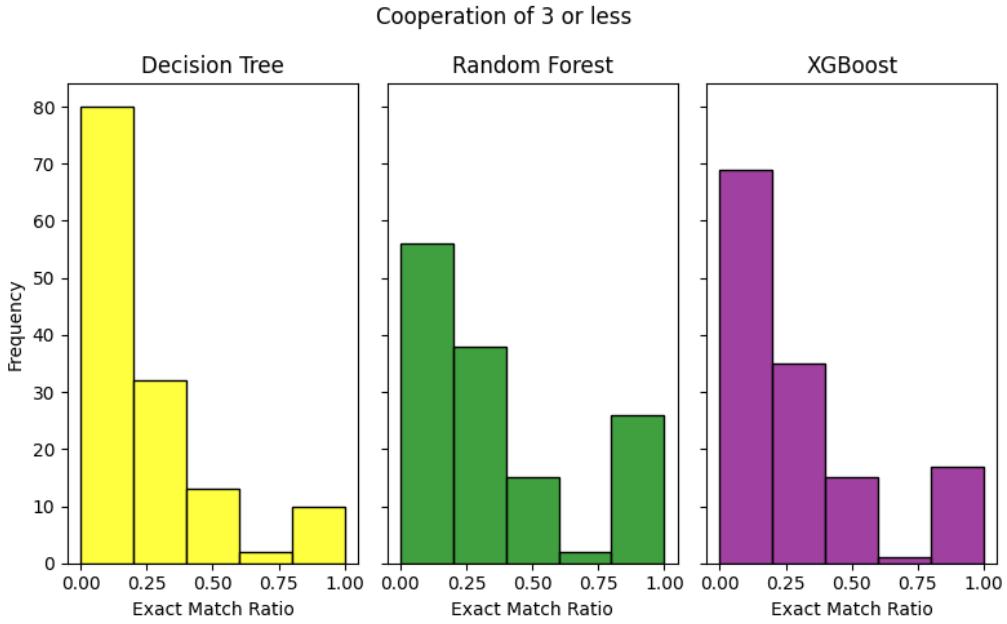

Figure 4.23: Exact match ratio for all models. Cooperation of 3.

of collaborations. However, when larger collaborations are permitted, this effectiveness diminishes. In those instances, the suggested combinations of collaborations deviate more significantly from the predicted best ones.

Figures 4.25, 4.26 and 4.27 illustrate the error for all algorithms across different collab-

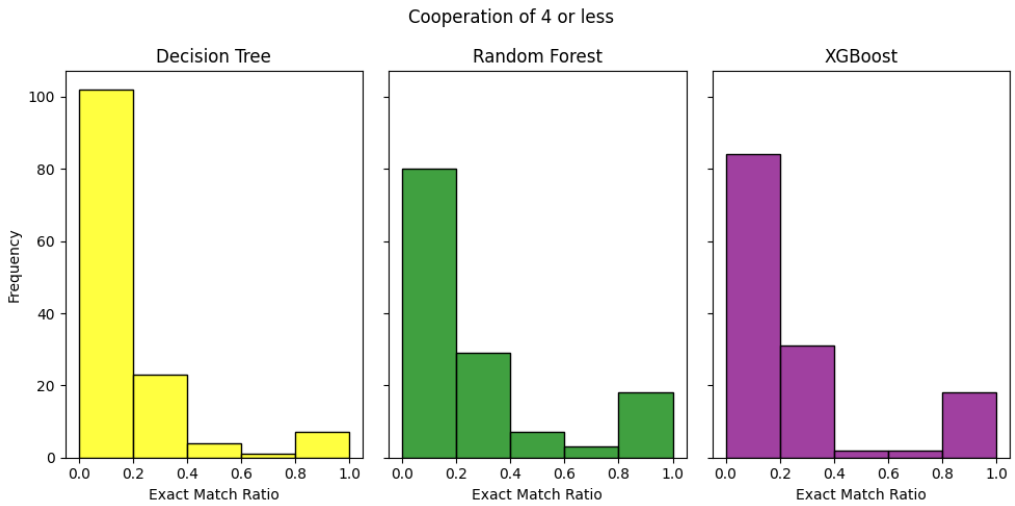

Figure 4.24: Exact match ratio for all models. Cooperation of 4.

oration sizes.

	Decision Tree		Random Forest		XGBoost			
Coop	Avg	Std	Avg	Std	Avg	Std	Min	Max
Two	92.19	171.09	77.77	152.33	117.01	184.67	270.0	7912.0
Three	2412.03	2193.10	2198.51	2392.35	2372.06	2165.56	304.0	10846.0
Four	2762.69	2461.96	2504.24	2471.44	2491.29	2293.46	336.0	11890.0

Table 4.11: Total gain difference: Decision Tree, Random Forest and XGBoost.

	Decision Tree		Random Forest		XGBoost	
Coop	Avg	Std	Avg	Std	Avg	Std
Two	0.03	0.05	0.03	0.05	0.04	0.06
Three	0.65	0.25	0.59	0.25	0.59	0.24
Four	0.68	0.23	0.59	0.26	0.61	0.24

Table 4.12: Total gain difference, percentage gap: Decision Tree, Random Forest and XGBoost.

Table 4.12 displays the average percentage gap from the true optimal solution. It is evident that for small collaboration tasks involving two or fewer companies, the algorithm consistently finds very good solutions, with average results very close to the actual best gains. This indicates that although the algorithm may not always identify the optimal assignments, it frequently finds very good ones. However, for larger collaborations involv-

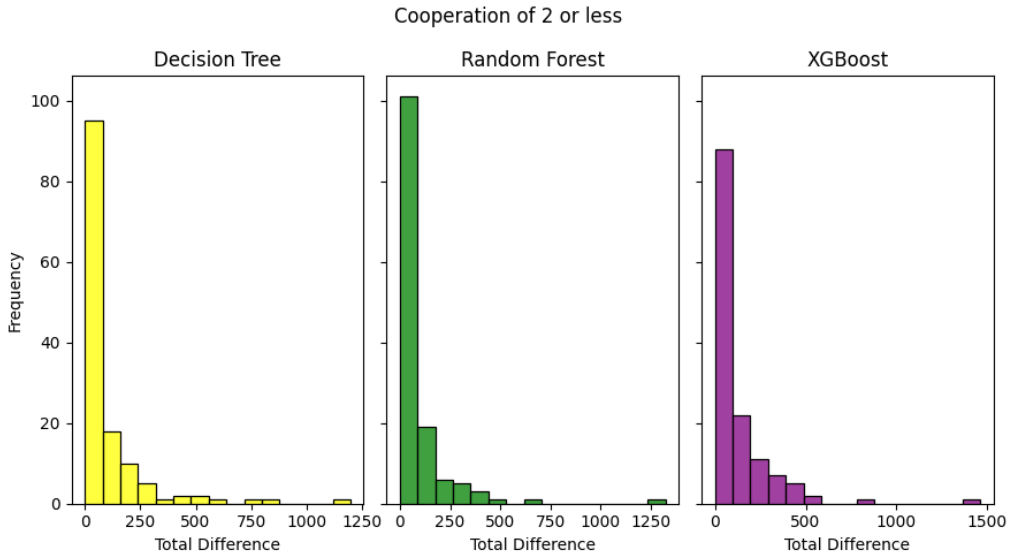

Figure 4.25: Total gain difference for all models. Cooperation of 2.

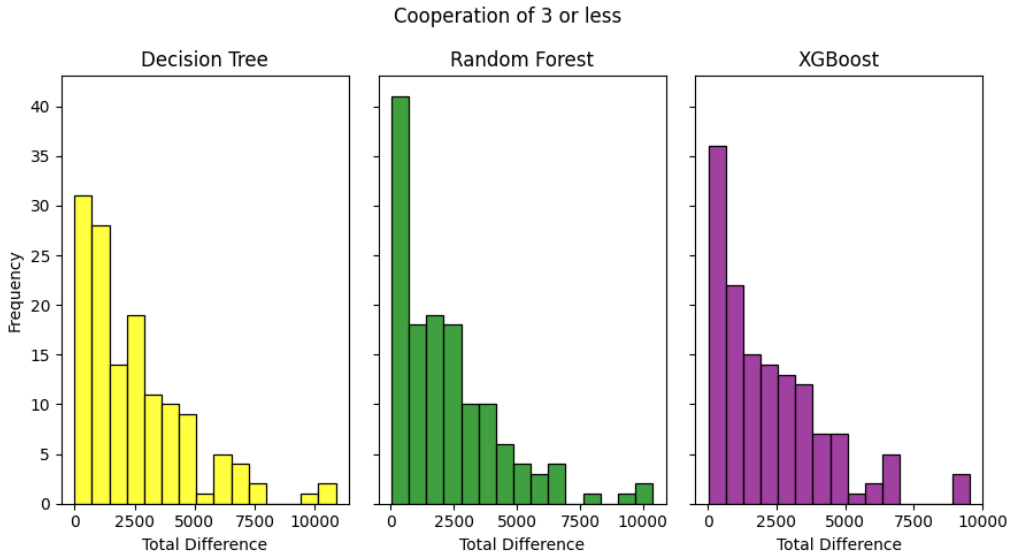

Figure 4.26: Total gain difference for all models. Cooperation of 3.

ing three or four companies, the algorithm's performance decreases significantly, with an average deviation of about 60% from the true solution. This suggests that the algorithm's suggested collaborations are often far from the optimal solutions, leading to significantly suboptimal assignments.

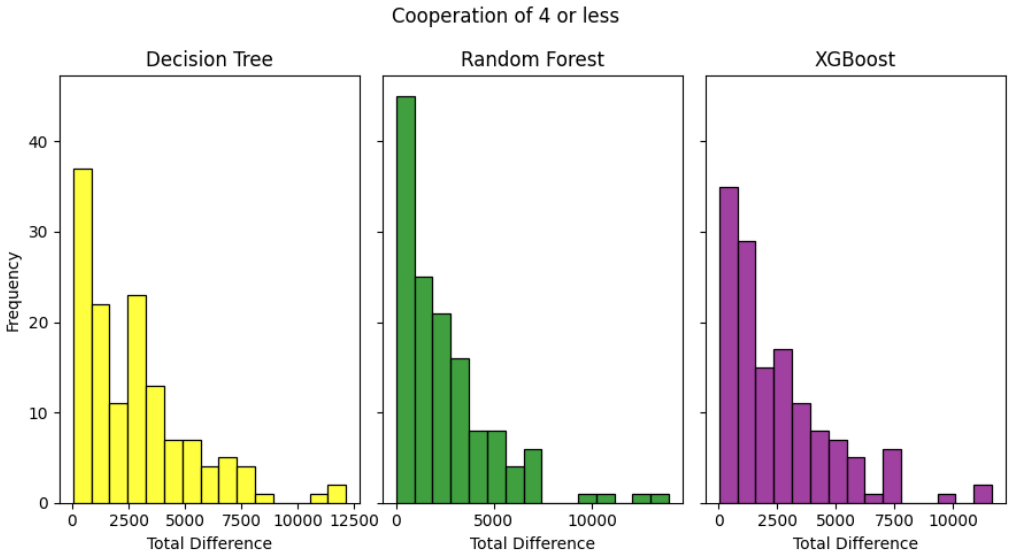

Figure 4.27: Total gain difference for all models. Cooperation of 4.

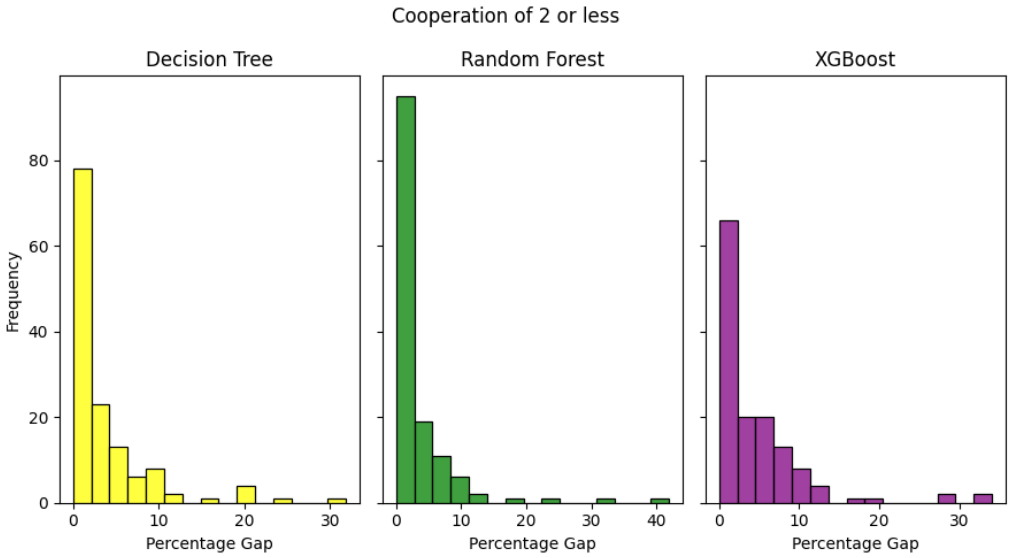

Figure 4.28: Total gain difference, percentage gap, for all models. Cooperation of 2.

4.2 Scaled up instances

The algorithm is only trained on instances with characteristics outlined in table 4.1, which includes a maximum of 20 companies and 800 deliveries. This limitation is due to the necessity of obtaining exact performance measures for the machine learning algorithms, requiring exact solutions. As shown in figure 4.11, the computational time to achieve

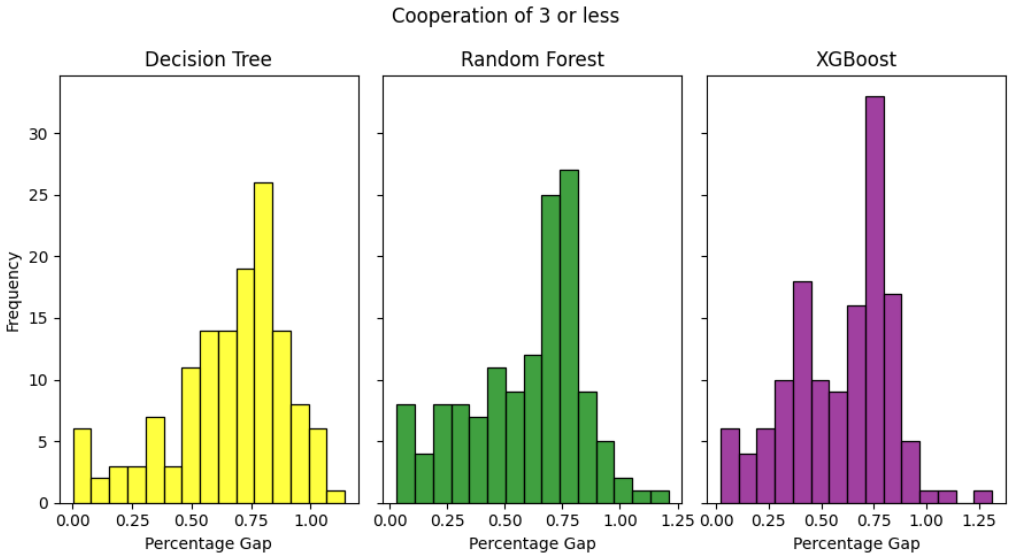

Figure 4.29: Total gain difference, percentage gap, for all models. Cooperation of 3.

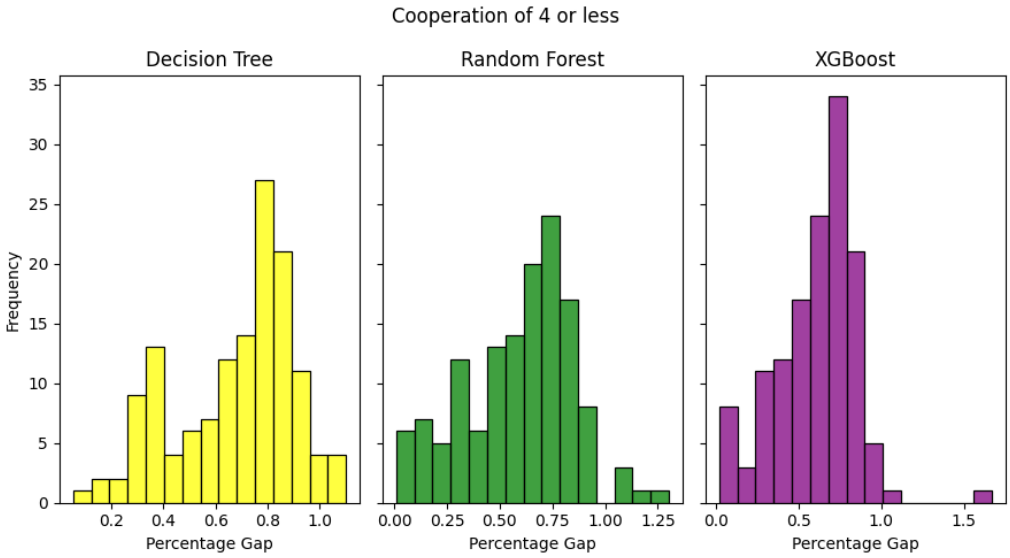

Figure 4.30: Total gain difference, percentage gap, for all models. Cooperation of 4.

exact solutions increases rapidly, making it impractical for larger instances. Consequently, the training instances were kept smaller. However, it remains interesting to evaluate how the algorithm performs on larger instances, both including more companies and more deliveries.

While current results are based on benchmark instances, it is also important to evaluate the algorithm's performance on larger instances. The primary reason for training the

algorithm on smaller instances is the need for true values in supervised machine learning. Obtaining these true values is time-consuming, and since vehicle routing problems are NP-hard, solving them is slow, limiting training to smaller instances. However, making predictions with the trained models is very quick, enabling the creation and evaluation of larger instances. To evaluate the algorithm's performance on these larger instances, the highest collaboration predictions will be examined along with a review of 20 random collaborations. The exact solution will be obtained by solving a vehicle routing problem for these predicted values to verify if they represent good collaborations.

The instances The instances that will be used are the *homberger_1000_customer_instances/RC1_10_1.TXT* and *homberger_800_customer_instances/RC1_8_1.TXT*. The first three scaled up instances utilize the *homberger_1000_customer_instances/RC1_10_1.TXT* file, while the second three scaled up instances utilize both of these instances combined. Table 4.13 shows the details about the instances used.

Idx	Instance	Deliveries	Capacity	Fixed cost	Owners	Depot
1	RC1_10_1_1000	1000	20	400	20	[300, 300]
2	RC1_10_1_1000	1000	20	400	30	[300, 300]
3	RC1_10_1_1000	1000	20	400	40	[300, 300]
4	RC1_10_1_1800	1800	20	400	20	[300, 300]
5	RC1_10_1_1800	1800	20	400	30	[300, 300]
6	RC1_10_1_1800	1800	20	400	40	[300, 300]

Table 4.13: An overview of the scaled up instances and their characteristics.

Performance metrics It is important to note that only the highest predictions can be compared to their true values in these cases, making it impossible to determine how the top collaborations rank against all other collaborations. Therefore, RMSE and R^2 will only be used to measure the error of these highest predictions. Additionally, precision@K or precision@K%, can not be utilized as the true ranking of the collaborations are unknown. However, it is possible to compare the ranking within those highest predicted results, to get a feeling of how well the algorithm manages to predict the gains relatively to the other gains. This can be done by looking at the performance of these instances in terms of the DTA.

Results The results look at the highest 10 collaborations. As it can be seen from the results on the benchmark instances, the predictions become less accurate with each step of the iterative method. This is due to the fact that the most important features in predicting the best cooperative partners of three and four are based on the prediction for cooperative partners of two. For that reason, this section will only focus on performance of the first step of the iterative method. The performance of this step sets the standard of how well the algorithm performs, and it can be expected to only do worse or similarly good in the iterative steps.

From the results in tables 4.14 and 4.15 it can be seen that the model predicts way lower value than the actual gains, despite the *VRP between clusters* features being able to estimate the expected gains quite accurately. This can very likely be due to the training data predominantly consisting of lower values. The model may have learned to predict lower values because it hasn't been exposed to enough higher values, leading the model to be biased towards predicting lower values.

However, to compensate for the unseen prediction values, the percentage gain is also considered as a prediction. This means that instead of predicting the exact expected gains, the increased benefits are considered. As mentioned in 3.4, using the percentage gains should be done carefully, as the fixed cost associated with the collaboration is also scaled with the total separate cost. Therefore it will only be used here to see if it has positive impact on the predictions. Table 4.14 shows the performance, when the percentage gains are utilized. However, it can be seen that using the prediction gains does not improve the results, but in fact makes them worse.

	XGBoost			**Random Forest**			**Decision Tree**		
Idx	*RMSE*	R^2	*DTA*	*RMSE*	R^2	*DTA*	*RMSE*	R^2	*DTA*
1	0.09	-4.85	0.47	0.03	0.02	0.51	0.05	-1.26	0.0
2	0.08	-2.09	0.60	0.03	0.16	0.64	0.02	0.10	0.13
3	0.10	-2.66	0.69	0.05	-0.86	0.44	0.03	-0.32	0.0
4	0.03	-2.78	0.44	0.04	-14.01	0.47	0.08	-10.99	0.0
5	0.09	-41.41	0.6	0.02	-0.84	0.76	0.06	-6.11	0.0
6	0.09	-21.73	0.56	0.01	-0.18	0.53	0.04	-1.99	0.0

Table 4.14: RMSE, R^2, and DTA for the scaled up instances where prediction is %gains.

Table 4.14 provide information on how well the models perform on scaled up instances in terms of percentage gains. It can be seen that in almost all cases the DTA using the Decision Tree model, is 0.0. This means that the model predicts all values as the same. Similarly

	XGBoost			Random Forest			Decision Tree		
Idx	*RMSE*	R^2	*DTA*	*RMSE*	R^2	*DTA*	*RMSE*	R^2	*DTA*
1	759.77	-5.07	0.53	825.24	-22.74	0.49	243.45	0.44	0.56
2	460.46	-4.15	0.67	580.62	-5.03	0.58	168.97	0.21	0.40
3	667.13	-3.46	0.82	482.43	-3.33	0.93	323.76	-0.28	0.58
4	533.81	-13.06	0.51	276.44	-1.18	0.69	177.0	0.07	0.27
5	540.21	-10.10	0.58	243.68	-1.38	0.64	261.06	0.27	0.51
6	552.38	-8.59	0.62	305.58	-2.04	0.67	170.33	0.09	0.44

Table 4.15: RMSE, R^2, and DTA for the scaled up instances where prediction is exact gains.

in the table 4.15, with the number gains, the Decision Tree does not do a lot better than random. For the Decision Tree algorithm this might be due to the fact that the values for the features used in these scaled up instances are a bit larger than the model has seen so far and therefore all values fall into the same category as the highest value the algorithm has seen so far, leading them to all or almost all being the same. However, while the Decision Tree algorithm appears to face this issue, the Random Forest and XGBoost models do not seem to struggle as much. In these cases, issue might be that the top 10 collaborations are all similarly effective, resulting in very similar features. This could mean the differences in gains among these 10 collaborations are minimal, making it difficult for the model to distinguish between them. However, to evaluate the model's overall performance on the instances, another approach is to randomly select a few collaborations and compare their predicted values to the actual values. These randomly selected collaborations are more likely to represent different collaboration pairings, resulting in more varied features and greater differences in real gains.

Tables 4.16, 4.17 and 4.18 present the results from 20 independent runs for 20 randomly selected collaborations. The data reveals that all algorithms perform significantly better in predicting the values and their rankings compared to predicting the top 10 collaborations. Among them, Random Forest exhibits the lowest error and highest DTA score, indicating the most accurate results. These findings highlight that while the algorithm does very well at identifying promising collaborations, its performance diminishes when the differences between collaborations are minimal.

Cooperation of two (Random 20: 20 times)						
	Decision Tree					
	RMSE		***R^2***		***DTA***	
Idx	**Avg**	**Std**	**Avg**	**Std**	**Avg**	**Std**
1	235.64	19.40	0.34	0.21	0.75	0.05
2	153.17	24.98	0.72	0.08	0.79	0.04
3	135.08	20.40	0.78	0.05	0.80	0.04
4	218.20	35.53	0.55	0.15	0.78	0.05
5	212.07	33.52	0.49	0.18	0.77	0.05
6	179.70	30.07	0.61	0.21	0.78	0.04

Table 4.16: Performance on 20 random instances: Decision Tree.

Cooperation of two (Random 20: 20 times)						
	Random Forest					
	RMSE		***R^2***		***DTA***	
Idx	**Avg**	**Std**	**Avg**	**Std**	**Avg**	**Std**
1	236.44	25.28	0.23	0.267	0.82	0.05
2	113.03	29.98	0.84	0.07	0.91	0.04
3	105.94	40.14	0.86	0.06	0.90	0.03
4	235.45	39.24	0.49	0.16	0.85	0.04
5	183.10	22.00	0.58	0.12	0.84	0.034
6	147.09	25.34	0.76	0.08	0.89	0.03

Table 4.17: Performance on 20 random instances: Random Forest.

Cooperation of two (Random 20: 20 times)						
	XGBoost					
	RMSE		***R^2***		***DTA***	
Idx	**Avg**	**Std**	**Avg**	**Std**	**Avg**	**Std**
1	465.79	30.83	-1.15	0.81	0.81	0.05
2	243.33	32.60	0.28	0.25	0.89	0.02
3	281.08	29.31	0.04	0.33	0.90	0.04
4	531.71	35.29	-1.57	0.93	0.84	0.03
5	398.22	53.62	-0.78	0.46	0.80	0.05
6	351.88	32.35	-0.49	0.46	0.88	0.03

Table 4.18: Performance on 20 random instances: XGBoost.

4.3 Discussion

From the results is can be seen that the iterative algorithm shows a lot of potential, however there are still parts of the algorithm that need to be improved on. It can be seen that the performance on the benchmark instances are quite good for finding collaborations of two companies. However, with every step of the iterative method the performance decreases. It is also noteworthy to mention that the performance of the iterative steps strongly depend on the performance of earlier steps, this is because the most important features for the latter steps are created from the predictions of the previous steps. This means that to be able to increase the performance of the later steps, it is important to improve on the earlier steps as well.

However, it is worth mentioning that in the iterative steps, both for finding collaborations of three companies and collaborations of four companies, the most important features are not consistent between all machine learning algorithms. When predicting the best collaborations for four companies, for the XGBoost, the most important feature is the *lowest subcoalition*, while in comparison, the most important feature for the Random Forest, is the *average of all*. In both cases the importance of those features is around 30%. This might indicate that the features created from the previous predictions can be improved on as well, increasing the performance of the iterative method. From table 4.5 the performance of each step of the iterative method can be seen, both in terms of RMSE and R^2. Interestingly, Random forest, the best performing model out of the three chosen models, achieved a R^2 score of 0.72 for the first step. Even though this is in many context considered quite strong, it is still a score that could be improved on. Specially as the first step should be very good for the rest of the algorithm to perform well. A R^2 score of 0.72 indicates that approximately 72% of the variance in the dependent variable can be explained by the independent variables in the model. This suggests that the model has good predictive power, although there is still 28% of the variability in the data that is not explained by the model. Additionally, emphasising what was discussed earlier, the R^2 score decreases drastically with each iterative step. This indicates that the features used in the iterative steps can be improved on. However, for the R^2 score, it should be kept in mind that this score is highly influenced by the RMSE of the model. Higher RMSE, results in lower R^2. Even though this effects the results of the model, the main goal is not to minimise the RMSE, but to correctly predict the gains, relative to the other gains. This indicates that a R^2 score of 0.72, might be better than it looks. In contrast the precision@K and DTA measures show quite a good performance when correctly predicting the highest predicted

gains and the overall ranking. This shows that the most promising collaborations are in fact, in most cases, predicted to be the most promising ones. This indicates that, in most cases, the algorithm successfully identifies the most promising collaborations among the benchmark instances, demonstrating its ability to distinguish between highly promising and less promising collaborations.

In the assignment task focused on maximizing overall gains, the algorithm performs very well when only two or fewer companies are allowed to collaborate, consistently finding optimal or nearly optimal solutions. However, when larger collaborations are permitted, the algorithm's performance decreases significantly, resulting in increasingly suboptimal solutions.

While the algorithm effectively predicts general trends regarding which collaborations are favorable or unfavorable, it struggles to differentiate between collaborations that are similarly good or similarly bad. Finally, when evaluating the algorithm's performance on larger instances with more companies and/or deliveries, it again has difficulty distinguishing between similar cases but does very well at identifying overall trends.

5

Conclusion and future research

5.1 Conclusion

The aim of this study was to develop an algorithm capable of identifying long-term collaborative partners that would maximize the benefits of cooperation. This was achieved by generating features and utilizing these features in an iterative machine learning approach.

The data used in this study was limited to information about the owner, location, and quantity of deliveries. This information alone was deemed insufficient to identify patterns that would indicate which companies would benefit most from cooperation. To address this, features were created to capture patterns within the data. These features were developed using clustering algorithms, routing algorithms, boxing algorithms, and other methods. Once these features were established, three machine learning algorithms were tested. The first was a simple Decision Tree to determine if a basic model could capture the data patterns. The other two were ensemble learning methods: Random Forest and XGBoost. These machine learning methods were applied iteratively, using predictions from smaller cooperation sizes to create features for predicting larger cooperation sizes.

The results indicate that the iterative method was effective in distinguishing between promising and non-promising collaborations, particularly in terms of ranking. The method was quite successful in ranking collaborations from best to worst. However, when it came to predicting the exact gains expected from collaboration, the algorithm faced difficulties. This, in turn, made the assignment problem sensitive to these gain predictions. Thus, while the algorithm performed well in identifying the most promising collaborations, it struggled to differentiate between similarly promising collaborations. This lack of precision made it less robust in assigning companies to collaborations in a way that maximized overall gains, specially when allowing for collaborations of up to four companies collaborating. This

sensitivity was less clear when only assigning companies to collaborations of two, where the algorithm consistently found very good solutions, with average results very close to the actual best gains.

To summarize, the iterative method shows very promising results in identifying long-term collaborative partners, particularly in terms of ranking, for small collaborations of up to four companies. The method effectively distinguishes between promising and less promising collaborations, making it useful for pinpointing the most promising options from a large pool of options. Once these promising collaborations are identified, they can be solved to an exact solution, demonstrating the algorithm's suitability for identifying the best partnerships.

5.2 Future research

However, despite the promising results, the algorithm can still be improved. Future research could extend the algorithm to handle different scenarios where the assumptions of this study are relaxed. This could include the following extensions:

- Increased flexibility in the types of companies: Currently, the algorithm assumes that all companies share a single depot. Future research could explore the potential for finding cooperative opportunities among companies with different depots.
- Additionally, it would be beneficial to consider scenarios involving both pickups and deliveries. This includes cases where companies have multiple pickup and delivery locations, with the requirement that pickup locations should be visited before the corresponding drop-off locations.
- Incorporate a range of order sizes, as currently, all orders are of size one. As mentioned in section 2.1.4, Cuervo et al. [8] conducted a simulation study that analysed the effects of different characteristics. They discovered that the average order size is the most influential factor, with companies having different order sizes benefiting the most. Therefore, incorporating a range of order sizes could show interesting results. Additionally, while this study assumed that all companies have a similar number of orders, it would be valuable to explore the impact of uneven order distributions on the results.

Other areas for improvement include developing additional features or enhancing the existing ones. For instance, the "on-the-way compatibility" feature did not yield promising

results, likely because the boxing method required fixed box thicknesses. When there were many deliveries, too many fell within the boxes, and with numerous clusters, the boxes covered excessive areas, leading to an overestimation of deliveries considered "on the way." This hindered the ability to detect patterns for determining compatibility between companies. Consequently, more effort should be invested in identifying better features for the iterative method and improving how predictions from earlier steps are utilized in subsequent steps. Although the initial step is quite promising, its accuracy declines rapidly, indicating that those features might require additional attention.

Since the clustering step is crucial for most features in the algorithm, future research could explore how varying cluster sizes impact the results. Currently, clusters are limited to five to fifteen orders each. It would be valuable to investigate the effects of larger clusters or use a different approach, such as determining cluster radius based on a maximum distance rather than just the number of orders. This adjustment could improve the scalability of the problem, enabling the algorithm to handle larger and more densely clustered instances more efficiently.

Lastly, while this algorithm is designed to identify potential long-term collaborative partners, exploring methods for fair allocation of gains would be beneficial. Future research could consider incorporating cost allocation methods from game theory, such as compromise stability, to ensure equitable distribution of benefits among partners.

References

[1] Adenso-Díaz, B., S. Lozano, S. Garcia-Carbajal, and K. Smith-Miles (2014). Assessing partnership savings in horizontal cooperation by planning linked deliveries. *Transportation Research 66*, 268–279. 14, 15

[2] Agarwal, R. and Ergunand (2010). Network design and allocation mechanisms for carrier alliances in liner shipping. *Operations Research 58 (6)*. 16

[3] Creemers, S., G. Woumans, R. Boute, and J. Beliënc (2017). Tri-vizor uses an efficient algorithm to identify collaborative shipping opportunities. *Informs Journal on Applied Analytics 47(3)*, 195–277. 13, 14, 17, 37

[4] Cruijssen, F., O. Bräysy, W. D. Dullaert, H. Fleuren, and M. Salomon (2016). Joint route planning under varying market conditions. *International Journal of Physical Distribution Logistics Management 37(4)*. 11, 12, 58

[5] Cruijssen, F., M. Cools, and W. Dullaert (2007). Horizontal cooperation in logistics: Opportunities and impediments. *Transportation Research Part E 43 (2)*, 129–142. 8

[6] Cruijssen, F., W. Dullaert, and H. Fleuren (2009). Horizontal cooperation in transport and logistics: A literature review. *Transportation Journal 46(3)*, 22–39. 1

[7] Cruijssen, F. and M. Salomon (2004, 01). Empirical study: Order sharing between transportation companies may result in cost reductions between 5 to 15 percent. *Tilburg University, Center for Economic Research, Discussion Paper 80*. 2

[8] Cuervo, D. P., C. Vanovermeire, and K. Sörensen (2016). Determining collaborative profits in coalitions formed by two partners with varying characteristics. *Transportation Research 70*, 171–184. 19, 44, 45, 97

[9] Doherty, S. and S. Hoyle (2009). Supply chain decarbonization: The role of logistics and transport in reducing supply chain carbon emissions. *Report, World Economic Forum, Geneva, Switzerland.*. 1

[10] Dondo, R. and J. Cerdá (2017). A cluster-based optimization approach for the multi-depot heterogeneous fleet vehicle routing problem with time windows. *European Journal of Operational Research 176*, 1478–1507. 17, 18, 33, 58

[11] European Commission (2019). Commission implementing regulation (eu) 2021/392 of 4 march 2021 on the monitoring and reporting of data relating to co2 emissions from passenger cars and light commercial vehicles pursuant to regulation (eu) 2019/631 of the european parliament and of the council and repealing commission implementing regulations (eu) no 1014/2010, (eu) no 293/2012, (eu) 2017/1152 and (eu) 2017/1153 (text with eea relevance). *Official Journal of the European Union*. 2

[12] European Commission (2021). Eu transport in figures – statistical pocketbook 2021. *Directorate-General for Mobility and Transport*. 2

[13] Gabiole, T. V. d. and A. Farrag-Thibault (2023). Road freight zero: Towards a holistic regulatory framework for reducing road freight emissions in europe. *Report, World Economic Forum, Geneva, Switzerland*. 1

[14] Gehring, H. and J. Homberger (2008). Gehring & Homberger benchmark. https://www.sintef.no/projectweb/top/vrptw/homberger-benchmark/. Retrieved August 2nd, 2024. 58

[15] Houghtalen, L., Ergun, and J. Sokol (2011). Designing mechanisms for the management of carrier alliances. *Transportation Science 45 (4)*, 465–482. 17

[16] Krajewska, M. and H. Kopfer (2006). Collaborating freight forwarding enterprises: Request allocation and profit sharing. *In: Kim, K.H., Günther, HO. (eds) Container Terminals and Cargo Systems*, 365–381. 10

[17] Li, J., Y. Fang, and N. Tang (2022, 05). A cluster-based optimization framework for vehicle routing problem with workload balance. *Computers Industrial Engineering 169*, 108221. 19, 33

[18] Li, X., P. Zhang, and G. Zhu (2019, 09). Dbscan clustering algorithms for non-uniform density data and its application in urban rail passenger aggregation distribution. *Energies 12*, 3722. 19

[19] Liu, F. and S. Shen (1999). The fleet size and mix vehicle routing problem with time windows. *Journal of the Operational Research Society 50*, 721–732. 11

[20] Ozener, O. and O. Ergun (2008, 05). Allocating costs in a collaborative transportation procurement network. *Transportation Science 42*, 146–165. 1, 4

[21] Pedregosa, F., G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research 12*, 2825–2830. 26, 33

[22] Song, J. and A. C. Regan (2004). An auction based collaborative carrier network. *Paper presented at the 83rd annual meeting of the Transportation Research Board (TRB), Washington, DC*. 9

[23] Verdonck, L., A. Caris, K. Ramaekers, and G. Janssens (2013, 10). Collaborative logistics from the perspective of road transportation companies. *Transport Reviews 33*, 700–719. 9, 10, 12, 16

[24] Winn, J. (2023). *Model-Based Machine Learning*. Chapman Hall. 46

Appendix

Decision Tree: Cooperation of 2	
Feature	**Importance**
VRP between clusters cost before and after cooperation with fixed cost	0.808053
VRP between clusters cost before and after cooperation	0.109863
Coalition cost_x	0.026223
Total cost before collaboration	0.014651
Silhouette score	0.013328
Instance Number	0.007017
Number of clusters	0.006484
Percentage Route Sharing	0.002465
Average distance to depot	0.002346
VRP between clusters	0.001943
Total truck utilization	0.001737
Percentage Route Sharing*Truck Utilization	0.001645
VRP between clusters cost per company	0.001428
KmeansScoreBinary	0.001316
DBSCANscore	0.000560
Truck capacity	0.000376
KmeansScore	0.000298
Trucks saved based on clusters	0.000132
Total number of deliveries	0.000068
Trucks Saved	0.000066
Number of clusters needing more than a truck	0.000000
Percentage of clusters needing more than a truck	0.000000
Coalition Size_x	0.000000

Table 5.1: Feature importance for cooperation of two: Decision Tree.

Decision Tree: Cooperation of 3	
Feature	**Importance**
Average of highest 2 subcoalitions	0.755021
Average of all	0.148667
Truck capacity	0.025073
Total cost before collaboration	0.020973
Number of clusters	0.009224
Total truck utilization	0.007056
Instance Number	0.006434
Trucks Saved	0.005531
Silhouette score	0.005025
Average distance to depot	0.004200
Coalition cost_x	0.002570
KmeansScore	0.002204
Average of highest 2 ranks	0.001514
Highest subcoalition	0.001270
KmeansScoreBinary	0.001193
Rank of Subcoalition 1	0.001080
Trucks saved based on clusters	0.000496
Rank of Subcoalition 2	0.000466
DBSCANscore	0.000439
Subcoalition 3	0.000382
Total number of deliveries	0.000320
Percentage Route Sharing	0.000285
Subcoalition 1	0.000242
Rank of Subcoalition 3	0.000229
Subcoalition 2	0.000105
Percentage Route Sharing*Truck Utilization	0.000000
Percentage of clusters needing more than a truck	0.000000
Coalition Size_x	0.000000
Number of clusters needing more than a truck	0.000000

Table 5.2: Feature importance for cooperation of three: Decision Tree

Decision Tree: Cooperation of 4	
Feature	**Importance**
Average of highest 2 subcoalitions	0.586428
Average of all	0.291883
Lowest subcoalition	0.034416
Total cost before collaboration	0.026771
Coalition cost_x	0.018345
Instance Number	0.016522
Average distance to depot	0.008685
Number of clusters	0.007025
Average of highest 3 subcoalitions	0.003011
Highest subcoalition	0.001728
Truck capacity	0.001515
Rank of Subcoalition 1	0.001220
Total number of deliveries	0.000954
Rank of Subcoalition 2	0.000810
Rank of Subcoalition 3	0.000533
Subcoalition 3	0.000156
Number of clusters needing more than a truck	0.000000
Coalition Size_x	0.000000
Percentage Route Sharing	0.000000
Percentage Route Sharing*Truck Utilization	0.000000
Silhouette score	0.000000
KmeansScore	0.000000
Subcoalition 1	0.000000
Subcoalition 2	0.000000
Subcoalition 4	0.000000
Percentage of clusters needing more than a truck	0.000000
Total truck utilization	0.000000
Trucks saved based on clusters	0.000000
Trucks Saved	0.000000
DBSCANscore	0.000000
Rank of Subcoalition 4	0.000000
KmeansScoreBinary	0.000000
Average of highest 2 ranks	0.000000
Average of highest 3 ranks	0.000000

Table 5.3: Feature importance for various features.

Random Forest: Cooperation of 2	
Feature	**Importance**
VRP between clusters cost before and after cooperation with fixed cost	0.688018
VRP between clusters cost before and after cooperation	0.164710
Total cost before collaboration	0.054529
Silhouette score	0.014182
Average distance to depot	0.011681
Coalition cost	0.009187
Number of clusters	0.007733
Instance Number	0.007121
Percentage Route Sharing	0.006910
VRP between clusters cost per company	0.006370
VRP between clusters	0.005609
Percentage Route Sharing*Truck Utilization	0.005104
KmeansScore	0.003727
Total truck utilization	0.003669
DBSCANscore	0.002930
KmeansScoreBinary	0.002706
Truck capacity	0.002456
Total number of deliveries	0.001375
Trucks Saved	0.001001
Trucks saved based on clusters	0.000785
Number of clusters needing more than a truck	0.000111
Percentage of clusters needing more than a truck	0.000086
Coalition Size	0.000000

Table 5.4: Feature importance for cooperation of two: Random Forest.

Random Forest: Cooperation of 3	
Feature	**Importance**
Average of all	0.387368
Average of highest 2 subcoalitions	0.171554
Truck capacity	0.094812
Total cost before collaboration	0.087306
Highest subcoalition	0.048073
Number of clusters	0.025730
Instance Number	0.024820
DBSCANscore	0.019987
Average distance to depot	0.017857
Silhouette score	0.013546
Total truck utilization	0.013470
KmeansScoreBinary	0.012884
Subcoalition 3	0.012233
KmeansScore	0.012233
Average of highest 2 ranks	0.009467
Subcoalition 1	0.005712
Rank of Subcoalition 3	0.005189
Coalition cost	0.004875
Rank of Subcoalition 2	0.004826
Rank of Subcoalition 1	0.004783
Subcoalition 2	0.004611
Percentage Route Sharing*Truck Utilization	0.004092
Total number of deliveries	0.004066
Trucks Saved	0.004046
Percentage Route Sharing	0.003366
Trucks saved based on clusters	0.001905
Number of clusters needing more than a truck	0.000823
Percentage of clusters needing more than a truck	0.000779
Coalition Size	0.000000

Table 5.5: Feature importance for cooperation of two: Random Forest.

Random Forest: Cooperation of 4	
Feature	**Importance**
Average of all	0.345104
Total cost before collaboration	0.149962
Average of highest 2 subcoalitions	0.111131
Average of highest 3 subcoalitions	0.106839
Highest subcoalition	0.084894
Lowest subcoalition	0.037897
Average distance to depot	0.030546
Number of clusters	0.018971
Coalition cost	0.018576
Rank of Subcoalition 1	0.016333
Average of highest 2 ranks	0.008862
Instance Number	0.008089
Subcoalition 1	0.007071
Rank of Subcoalition 4	0.006956
Rank of Subcoalition 2	0.006117
Rank of Subcoalition 3	0.005984
Subcoalition 2	0.005666
Subcoalition 3	0.005600
Average of highest 3 ranks	0.005496
Number of clusters needing more than a truck	0.005427
Subcoalition 4	0.004839
Percentage of clusters needing more than a truck	0.004374
Truck capacity	0.003083
Total number of deliveries	0.002183
DBSCANscore	0.000000
Total truck utilization	0.000000
Trucks saved based on clusters	0.000000
Trucks Saved	0.000000
KmeansScore	0.000000
Silhouette score	0.000000
Percentage Route Sharing*Truck Utilization	0.000000
Percentage Route Sharing	0.000000
Coalition Size	0.000000
KmeansScoreBinary	0.000000

Table 5.6: Feature importance for cooperation of four: Random Forest.

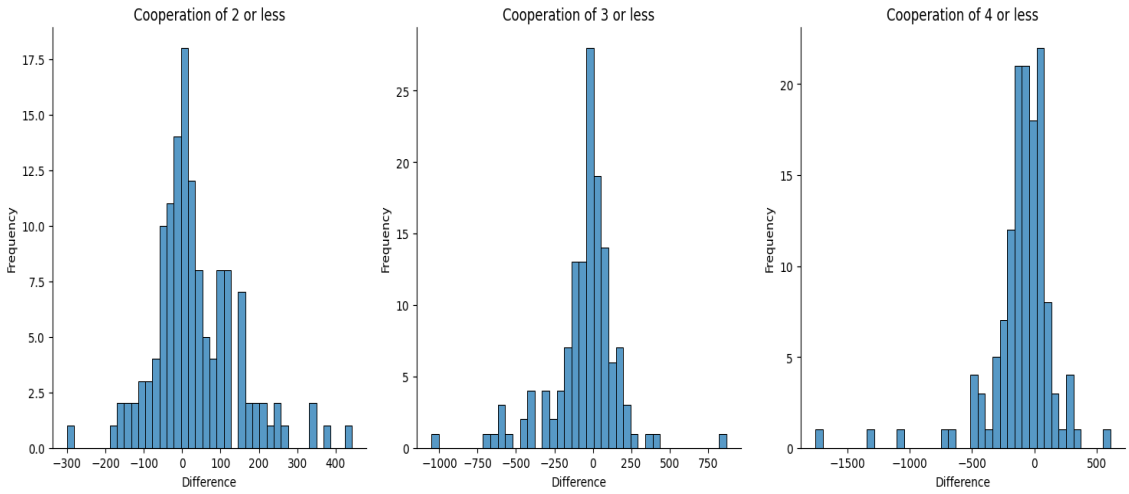

Figure 5.1: Decision Tree: Difference between the true and the predicted total gains, allowing for cooperation of two to four companies or less.

	Abs Average	*Abs Std*	*Min*	*Max*
Cooperation of two	80.6173	82.797	270.0	7912.0
Cooperation of three	147.242	174.722	304.0	10846.0
Cooperation of four	170.781	230.886	336.0	11890.0

Table 5.7: Total gains difference: Decision Tree.

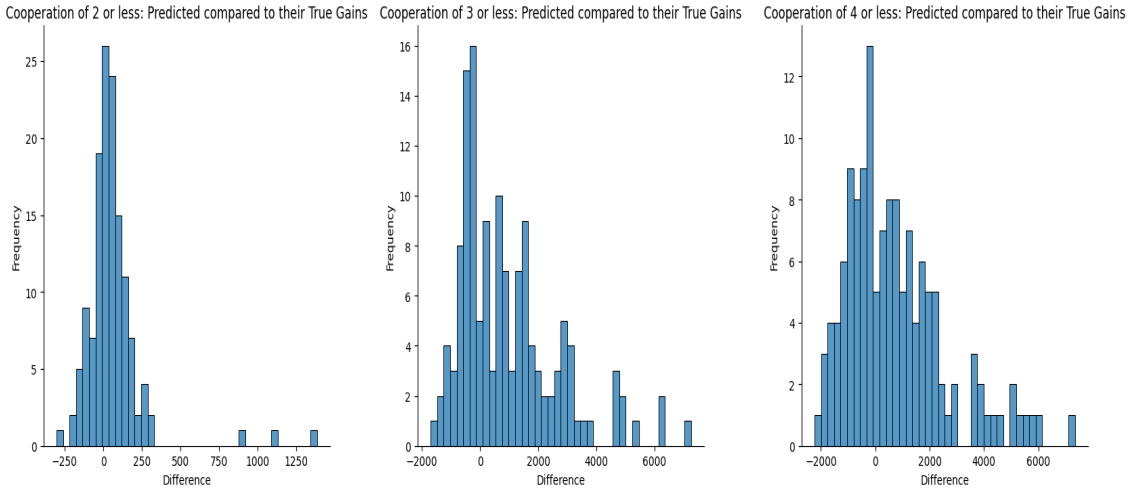

Figure 5.2: Decision Tree: Difference between the true gains of predicted assignment and the predicted gains of the predicted assignment. Cooperation of two to four companies.

	Abs Average	*Abs Std*	*Min*	*Max*
Cooperation of two	109.067	171.739	270.0	7912.0
Cooperation of three	1364.887	1403.920	304.0	10846.0
Cooperation of four	1423.421	1386.880	336.0	11890.0

Table 5.8: Total gains difference between the true gains and the the predicted gains of the predicted assignment: Decision tree.

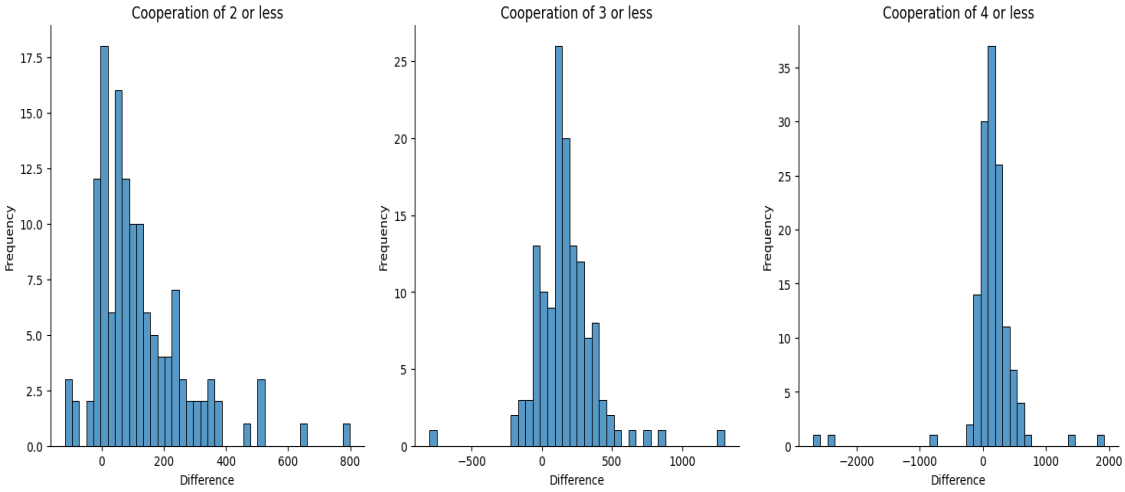

Figure 5.3: Random Forest: Difference between the true and the predicted total gains, allowing for cooperation of two to four companies or less.

	Abs Average	*Std*	*Min*	*Max*
Cooperation of two	131.018	137.182	270.0	7912.0
Cooperation of three	202.915	188.304	304.0	10846.0
Cooperation of four	246.459	364.827	336.0	11890.0

Table 5.9: Total gains difference: Random Forest.

	Average	*Std*
Cooperation of two	0.594	0.342
Cooperation of three	0.253	0.333
Cooperation of four	0.242	0.345

Table 5.10: Exact Match Ratio: Random Forest.

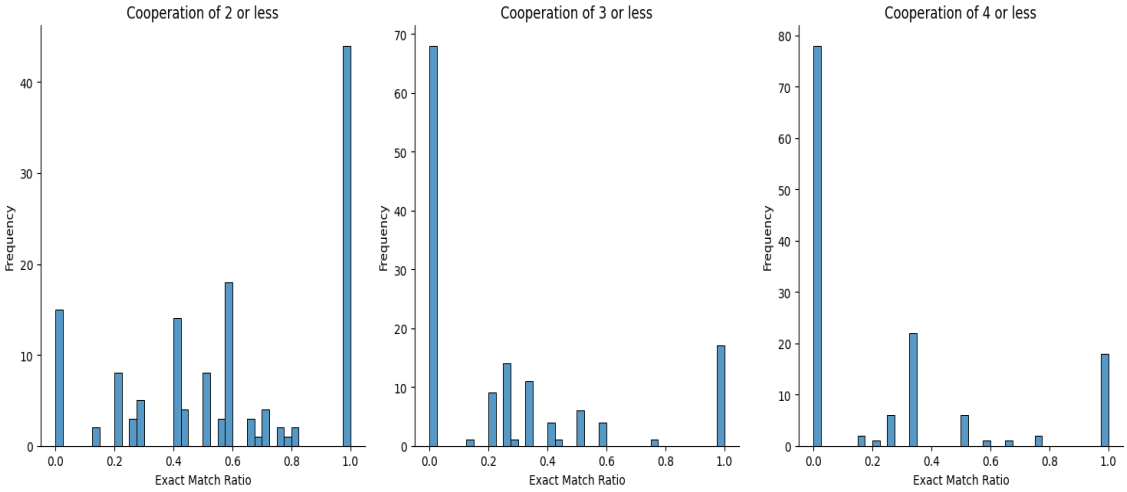

Figure 5.4: Random Forest: Exact Match Ratio for cooperation of two to four companies or less.

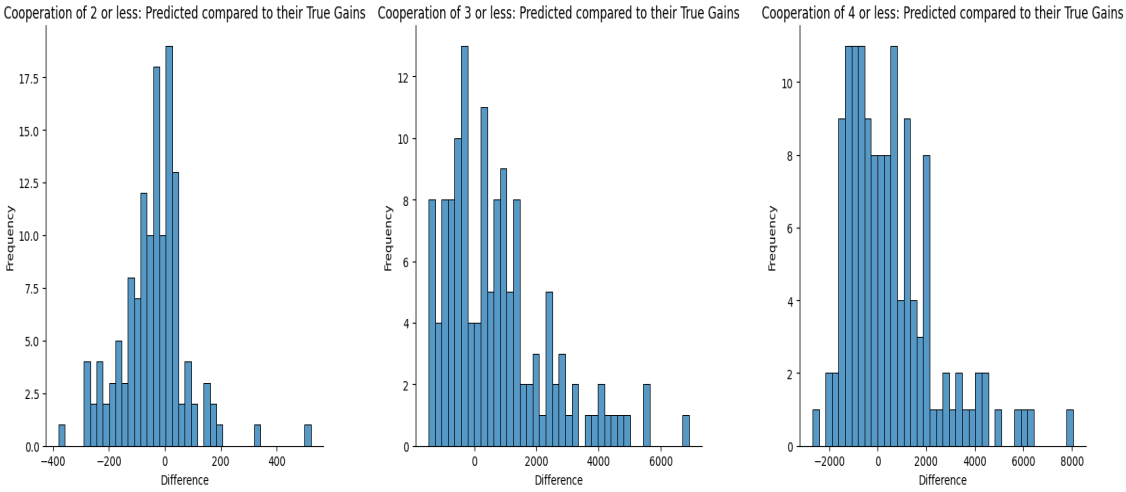

Figure 5.5: Random Forest: Difference between the true gains of predicted assignment and the predicted gains of the predicted assignment. Cooperation of two to four.

	Abs Average	*Abs Std*	*Min*	*Max*
Cooperation of two	89.221	88.095	270.0	7912.0
Cooperation of three	1294.416	1265.959	304.0	10846.0
Cooperation of four	1369.849	1346.573	336.0	11890.0

Table 5.11: Total gains difference between the true gains and the the predicted gains of the predicted assignment: Random forest.

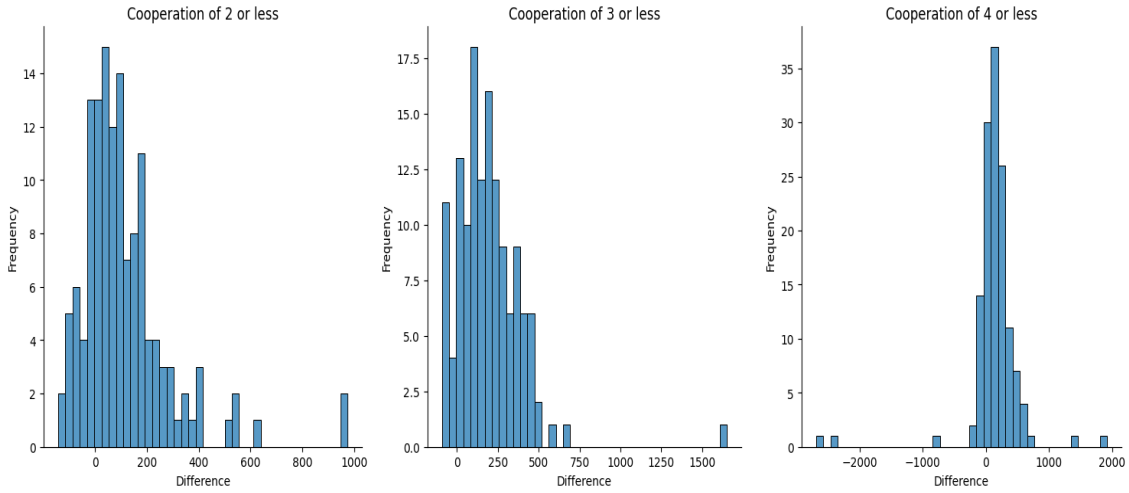

Figure 5.6: XGBoost: Difference between the true and the predicted total gains, allowing for cooperation of two to four companies or less.

	Abs Average	*Std*	*Min*	*Max*
Cooperation of two	136.794	156.283	270.0	7912.0
Cooperation of three	202.915	188.304	304.0	10846.0
Cooperation of four	382.871	419.105	336.0	11890.0

Table 5.12: Total gains difference: XGBoost.

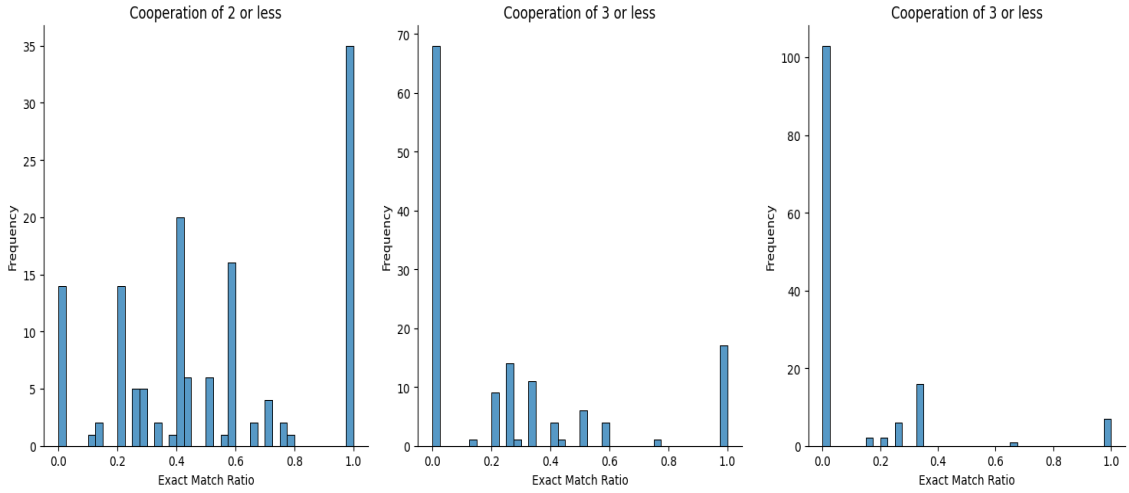

Figure 5.7: XGBoost: Exact Match Ratio for cooperation of two to four companies or less.

	Average	*Std*
Cooperation of two	0.527	0.334
Cooperation of three	0.253	0.333
Cooperation of four	0.111	0.242

Table 5.13: Exact Match Ratio: XGBoost.

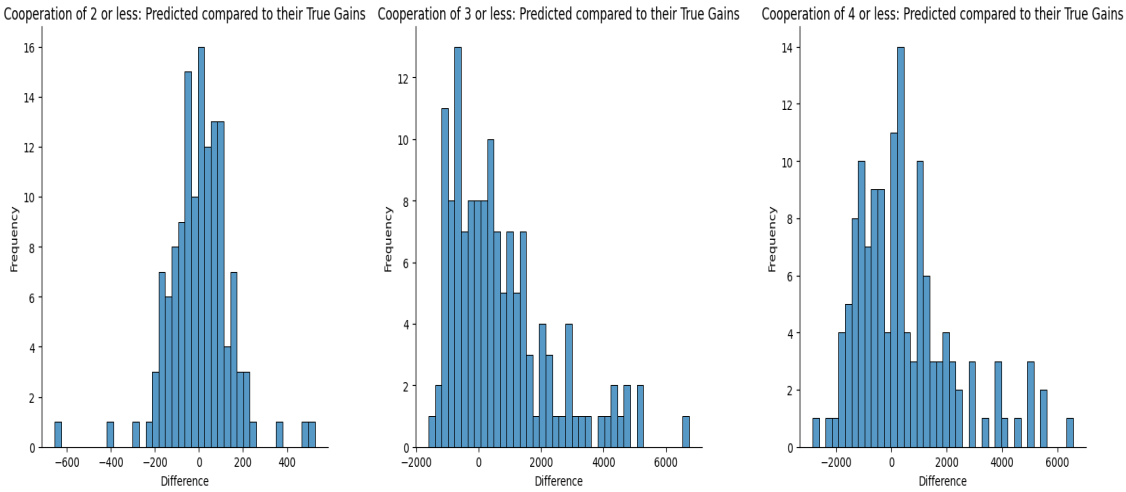

Figure 5.8: XGBoost: Difference between the true gains of predicted assignment and the predicted gains of the predicted assignment. Cooperation of two to four.

	Abs Average	*Abs Std*	*Min*	*Max*
Cooperation of two	101.349	98.621	270.0	7912.0
Cooperation of three	1233.743	1251.164	304.0	10846.0
Cooperation of four	1328.509	1261.070	336.0	11890.0

Table 5.14: Total gains difference between the true gains and the the predicted gains of the predicted assignment: XGBoost.

	Decision Tree		**Random Forest**		**XGBoost**			
Coop	Abs Avg	Abs Std	Abs Avg	Abs Std	Abs Avg	Abs Std	Min	Max
Two	80.6	82.8	131.0	137.2	136.8	156.3	270.0	7912.0
Three	147.2	174.7	202.9	188.3	202.9	188.3	304.0	10846.0
Four	170.8	230.9	246.5	364.8	382.9	419.1	336.0	11890.0

Table 5.15: Total gains difference: Decision Tree, Random Forest, and XGBoost.

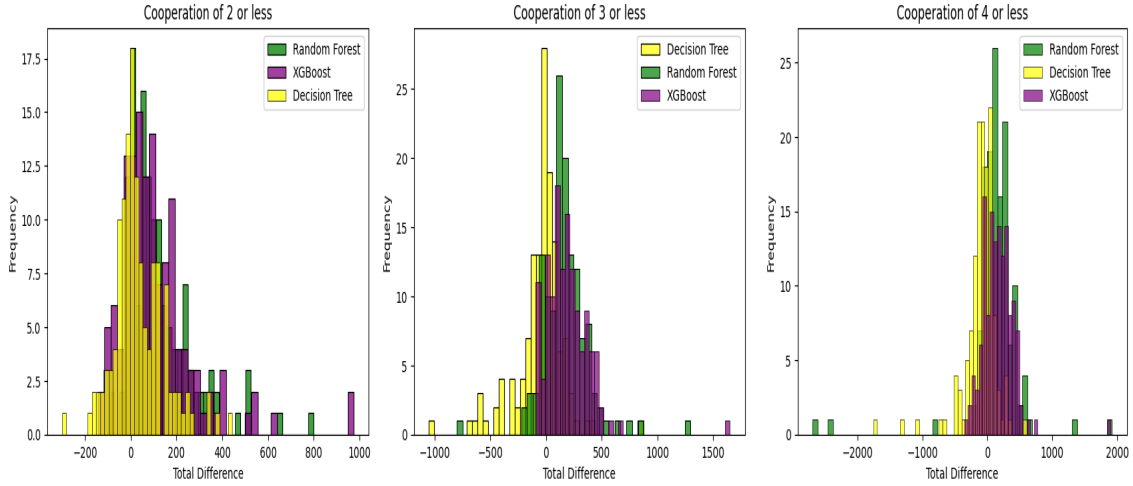

Figure 5.9: Difference between the true and the predicted total gains for different cooperation sizes.

	Decision Tree		Random Forest		XGBoost			
Coop.	Abs Avg	Abs Std	Abs Avg	Abs Std	Abs Avg	Abs Std	Min	Max
Two	109.1	171.7	89.2	88.1	101.4	98.6	270.0	7912.0
Three	1364.9	1403.9	1294.4	1266.0	1233.7	1251.2	304.0	10846.0
Four	1423.4	1386.9	1369.9	1346.6	1328.5	1261.1	336.0	11890.0

Table 5.16: Total gains difference between the true gains and the predicted gains of the predicted assignment: Decision Tree, Random Forest, and XGBoost.

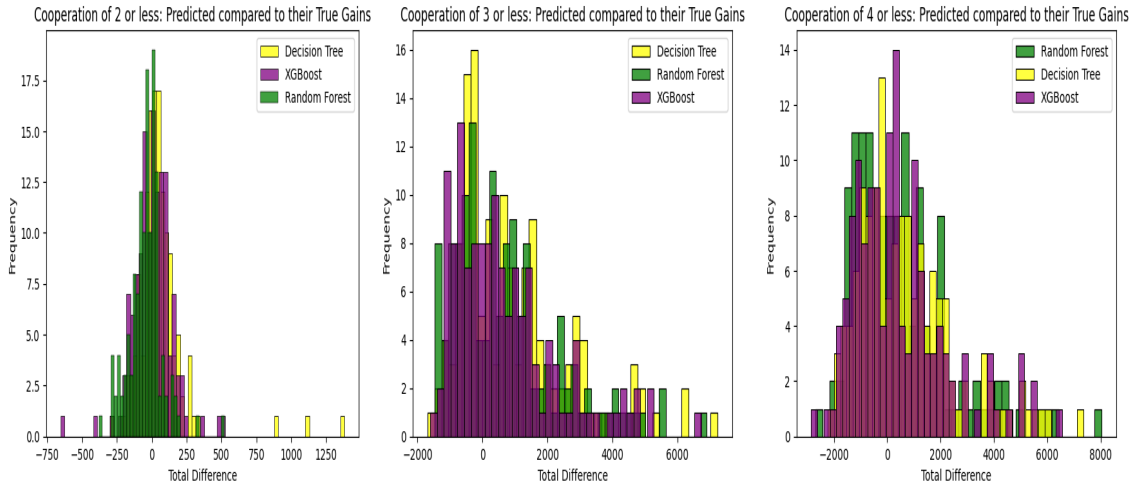

Figure 5.10: Difference between the true gains of predicted assignment and the predicted gains of the predicted assignment for different cooperation sizes.

Decision Tree: Untuned (Precision at K)						
	Cooperation 2		**Cooperation 3**		**Cooperation 4**	
	Average	*Std*	*Average*	*Std*	*Average*	*Std*
Top 10 in 10	0.634	0.195	0.367	0.212	0.256	0.188
Top 10 in 15	0.795	0.206	0.461	0.232	0.344	0.234
Top 10 in 20	0.871	0.187	0.549	0.255	0.429	0.263
Top 20 in 20	0.761	0.160	0.470	0.189	0.355	0.202

Table 5.17: How many of the best cooperation pairs are predicted as the best ones in Decition Tree (DT) iterative algorithm. Average of 138 instances.

Random Forest: Untuned (Precision at K)						
	Cooperation 2		**Cooperation 3**		**Cooperation 4**	
	Average	*Std*	*Average*	*Std*	*Average*	*Std*
Top 10 in 10	0.729	0.169	0.492	0.195	0.430	0.213
Top 10 in 15	0.810	0.139	0.594	0.182	0.502	0.198
Top 10 in 20	0.938	0.142	0.717	0.224	0.625	0.273
Top 20 in 20	0.876	0.163	0.638	0.223	0.540	0.250

Table 5.18: How many of the best cooperation pairs are predicted as the best ones in XG-Boost iterative algorithm. Average of 138 instances.

XG-Boost: Untuned (Precision at K)						
	Cooperation 2		**Cooperation 3**		**Cooperation 4**	
	Average	*Std*	*Average*	*Std*	*Average*	*Std*
Top 10 in 10	0.732	0.171	0.509	0.206	0.452	0.187
Top 10 in 15	0.872	0.172	0.638	0.225	0.577	0.229
Top 10 in 20	0.929	0.160	0.727	0.220	0.673	0.236
Top 20 in 20	0.820	0.141	0.607	0.184	0.547	0.185

Table 5.19: How many of the best cooperation pairs are predicted as the best ones in XG-Boost iterative algorithm. Average of 138 instances.

Decision Tree: Tuned (Precision at K)						
	Cooperation 2		**Cooperation 3**		**Cooperation 4**	
	Average	*Std*	*Average*	*Std*	*Average*	*Std*
Top 10 in 10	0.666	0.194	0.289	0.225	0.191	0.169
Top 10 in 15	0.816	0.184	0.396	0.272	0.242	0.206
Top 10 in 20	0.907	0.159	0.479	0.299	0.306	0.248
Top 20 in 20	0.789	0.143	0.394	0.218	0.245	0.208

Table 5.20: How many of the best cooperation pairs are predicted as the best ones in XG-Boost iterative algorithm. Average of 138 instances.

Random Forest: Tuned (Precision at K)						
	Cooperation 2		**Cooperation 3**		**Cooperation 4**	
	Average	*Std*	*Average*	*Std*	*Average*	*Std*
Top 10 in 10	0.738	0.176	0.505	0.202	0.456	0.189
Top 10 in 15	0.877	0.162	0.654	0.232	0.578	0.223
Top 10 in 20	0.943	0.135	0.732	0.230	0.671	0.239
Top 20 in 20	0.814	0.132	0.606	0.187	0.550	0.177

Table 5.21: How many of the best cooperation pairs are predicted as the best ones in XG-Boost iterative algorithm. Average of 138 instances.

XG-Boost: Tuned (Precision at K)						
	Cooperation 2		**Cooperation 3**		**Cooperation 4**	
	Average	*Std*	*Average*	*Std*	*Average*	*Std*
Top 10 in 10	0.748	0.160	0.516	0.183	0.463	0.203
Top 10 in 15	0.885	0.153	0.662	0.204	0.590	0.239
Top 10 in 20	0.944	0.135	0.753	0.210	0.682	0.258
Top 20 in 20	0.826	0.124	0.621	0.169	0.555	0.187

Table 5.22: How many of the best cooperation pairs are predicted as the best ones in XG-Boost iterative algorithm. Average of 138 instances. After tuning the hyperparameters.

Decision Tree: Tuned (Precision at K%)						
	Cooperation 2		**Cooperation 3**		**Cooperation 4**	
	Average	*Std*	*Average*	*Std*	*Average*	*Std*
Top 10% in 10%	0.588	0.249	0.371	0.217	0.333	0.215
Top 10% in 15%	0.760	0.244	0.500	0.253	0.473	0.259
Top 10% in 20%	0.821	0.223	0.614	0.267	0.594	0.266

Table 5.23: The precision at 10%. Average over all 138 instances.

Random Forest: Tuned (Precision at K%)						
	Cooperation 2		**Cooperation 3**		**Cooperation 4**	
	Average	*Std*	*Average*	*Std*	*Average*	*Std*
Top 10% in 10%	0.634	0.229	0.591	0.174	0.497	0.193
Top 10% in 15%	0.833	0.205	0.717	0.205	0.609	0.239
Top 10% in 20%	0.884	0.185	0.789	0.213	0.688	0.257

Table 5.24: The percision at 10%. Average over all 138 instances.

XGBoost: Tuned (Precision at K%)						
	Cooperation 2		**Cooperation 3**		**Cooperation 4**	
	Average	*Std*	*Average*	*Std*	*Average*	*Std*
Top 10% in 10%	0.668	0.217	0.595	0.160	0.626	0.147
Top 10% in 15%	0.847	0.200	0.751	0.159	0.777	0.167
Top 10% in 20%	0.902	0.177	0.835	0.152	0.852	0.156

Table 5.25: The precision at 10%. Average over all 138 instances.

Name	First inst.	#Instances	Capacity	Cost	Owners	Depot
800/C1_8_1	0	100	15	100	10	[100, 100]
800/C1_8_1	100	100	20	100	10	[100, 100]
800/C1_8_1	200	100	20	100	10	[100, 100]
800/C1_8_1	300	100	20	100	10	[100, 100]
800/C1_8_1	400	100	20	100	15	[100, 100]
800/C1_8_1	500	100	20	100	15	[100, 100]
800/C1_8_1	600	100	20	100	15	[100, 100]
800/C1_8_1	700	100	20	100	15	[100, 100]
800/R1_8_1	0	200	20	100	15	[100, 100]
800/R1_8_1	200	200	20	100	15	[100, 100]
800/R1_8_1	400	200	20	100	10	[100, 100]
800/R1_8_1	600	200	20	100	10	[100, 100]
800/C2_8_1	0	100	10	200	15	[100, 100]
800/C2_8_1	100	100	10	200	15	[100, 100]
800/C2_8_1	200	100	10	200	15	[100, 100]
800/C2_8_1	300	100	10	200	15	[100, 100]
800/C2_8_1	400	100	20	200	15	[100, 100]
800/C2_8_1	500	100	20	200	15	[100, 100]
800/C2_8_1	600	100	20	200	15	[100, 100]
800/C2_8_1	700	100	20	200	15	[100, 100]
800/R2_8_1	0	100	20	150	5	[100, 100]
800/R2_8_1	100	100	20	150	5	[100, 100]
800/R2_8_1	200	100	20	150	5	[100, 100]
800/R2_8_1	300	100	20	150	5	[100, 100]
800/R2_8_1	400	100	20	0	5	[100, 100]
800/R2_8_1	500	100	20	0	5	[100, 100]
800/R2_8_1	600	200	20	300	5	[100, 100]
800/RC2_8_1	0	50	15	200	10	[100, 100]
800/RC2_8_1	50	50	15	200	10	[100, 100]
800/RC2_8_1	100	50	15	200	10	[100, 100]
800/RC2_8_1	150	50	10	200	10	[100, 100]
800/RC2_8_1	200	50	10	200	10	[100, 100]
800/RC2_8_1	250	50	10	200	10	[100, 100]
800/RC2_8_1	300	50	10	0	10	[100, 100]
800/RC2_8_1	350	50	10	0	10	[100, 100]
800/RC2_8_1	400	50	10	0	10	[100, 100]
800/RC2_8_1	450	50	10	0	10	[100, 100]

Table 5.26: Table 1 of all benchmark instances. The instance names show *original instance size/used instance.*

Name	First inst.	#Instances	Capacity	Cost	Owners	Depot
800/RC2_8_1	500	100	10	0	10	[100, 100]
800/RC2_8_1	600	100	10	0	10	[100, 100]
800/RC2_8_1	700	100	10	100	10	[100, 100]
400/C1_4_1	0	50	10	0	5	[100, 100]
400/C1_4_1	50	50	20	0	5	[100, 100]
400/C1_4_1	100	50	20	0	5	[100, 100]
400/C1_4_1	150	50	20	0	5	[100, 100]
400/C1_4_1	200	100	20	0	5	[100, 100]
400/C1_4_1	300	100	20	0	5	[100, 100]
400/R1_4_1	0	100	20	0	10	[100, 100]
400/R1_4_1	100	100	20	50	10	[100, 100]
400/R1_4_1	200	100	20	50	10	[100, 100]
400/R1_4_1	300	100	20	50	10	[100, 100]
400/R2_4_1	0	200	30	100	10	[100, 100]
400/R2_4_1	200	200	30	100	10	[100, 100]
400/C2_4_1	0	200	30	100	10	[100, 100]
400/C2_4_1	200	200	30	100	10	[100, 100]
400/RC2_4_1	0	200	30	100	15	[100, 100]
400/RC2_4_1	200	200	30	100	15	[100, 100]
600/C1_6_1	0	50	15	200	10	[100, 100]
600/C1_6_1	50	50	15	200	10	[100, 100]
600/C1_6_1	100	100	15	200	10	[100, 100]
600/C1_6_1	200	100	15	200	10	[100, 100]
600/C1_6_1	300	100	15	200	10	[100, 100]
600/C1_6_1	400	100	15	200	10	[100, 100]
600/C1_6_1	500	100	15	200	10	[100, 100]
600/R1_6_1	0	200	20	100	10	[200, 200]
600/R1_6_1	200	200	20	100	10	[200, 200]
600/R1_6_1	400	200	20	100	10	[200, 200]
600/R2_6_1	0	200	30	100	10	[200, 200]
600/R2_6_1	200	200	30	100	10	[200, 200]
600/R2_6_1	400	100	30	100	10	[200, 200]
600/R2_6_1	500	100	30	100	10	[200, 200]
600/C2_6_1	0	100	30	100	10	[200, 200]
600/C2_6_1	100	100	30	100	10	[200, 200]
600/C2_6_1	200	100	20	100	10	[200, 200]

Table 5.27: Table 2 of all benchmark instances. The instance names show *original instance size/used instance.*

Name	First inst.	#Instances	Capacity	Cost	Owners	Depot
600/C2_6_1	300	100	20	100	10	[200, 200]
600/C2_6_1	400	100	15	100	10	[200, 200]
600/C2_6_1	500	100	15	100	10	[200, 200]
600/RC1_6_1	0	200	20	200	20	[200, 200]
600/RC1_6_1	200	200	20	200	20	[200, 200]
600/RC1_6_1	400	200	20	200	20	[200, 200]
1000/C1_10_1	0	200	30	0	10	[200, 200]
1000/C1_10_1	200	200	30	0	10	[200, 200]
1000/C1_10_1	400	200	30	0	10	[200, 200]
1000/C1_10_1	600	200	60	0	10	[200, 200]
1000/C1_10_1	800	200	60	0	10	[200, 200]
1000/R1_10_1	0	150	30	0	15	[200, 200]
1000/R1_10_1	150	150	30	0	15	[200, 200]
1000/R1_10_1	300	150	30	0	15	[200, 200]
1000/R1_10_1	450	150	20	0	15	[200, 200]
1000/R1_10_1	600	100	20	0	15	[200, 200]
1000/R1_10_1	700	100	20	200	15	[200, 200]
1000/R1_10_1	800	100	20	200	15	[200, 200]
1000/R1_10_1	900	100	20	200	15	[200, 200]
1000/RC1_10_1	0	100	20	200	5	[200, 200]
1000/RC1_10_1	100	100	20	200	5	[200, 200]
1000/RC1_10_1	200	100	10	200	5	[200, 200]
1000/RC1_10_1	300	100	10	200	5	[200, 200]
1000/RC1_10_1	400	100	30	200	10	[200, 200]
1000/RC1_10_1	500	100	30	200	10	[200, 200]
1000/RC1_10_1	600	100	30	200	10	[200, 200]
1000/RC1_10_1	700	100	30	0	10	[200, 200]
1000/RC1_10_1	800	100	20	0	10	[200, 200]
1000/RC1_10_1	900	100	20	0	10	[200, 200]
1000/R2_10_1	0	100	20	0	10	[400, 400]
1000/R2_10_1	100	100	20	0	10	[400, 400]
1000/R2_10_1	200	100	20	0	10	[400, 400]
1000/R2_10_1	300	100	20	50	10	[400, 400]
1000/R2_10_1	400	100	20	50	10	[400, 400]
1000/R2_10_1	500	100	20	50	10	[400, 400]
1000/R2_10_1	600	100	20	100	10	[400, 400]

Table 5.28: Table 3 of all benchmark instances. The instance names show *original instance size/used instance.*

Name	First inst.	#Instances	Capacity	Cost	Owners	Depot
1000/R2_10_1	700	100	20	100	10	[400, 400]
1000/R2_10_1	800	100	15	100	10	[400, 400]
1000/R2_10_1	900	100	15	100	10	[400, 400]
1000/C2_10_1	0	150	20	100	10	[400, 400]
1000/C2_10_1	150	150	20	100	10	[400, 400]
1000/C2_10_1	300	150	20	100	10	[400, 400]
1000/C2_10_1	450	150	20	100	10	[400, 400]
1000/C2_10_1	600	150	10	0	10	[400, 400]
1000/C2_10_1	750	150	10	0	10	[400, 400]
1000/C2_10_1	900	100	15	0	10	[400, 400]
200/C1_2_1	0	100	15	0	10	[100, 100]
200/C1_2_1	100	100	15	0	10	[100, 100]
200/C2_2_1	0	100	20	0	10	[100, 100]
200/C2_2_1	100	100	20	0	10	[100, 100]
200/R1_2_1	0	200	20	0	10	[100, 100]
200/R2_2_1	0	200	30	0	10	[100, 100]
200/RC2_2_1	0	50	15	0	5	[100, 100]
200/RC2_2_1	50	50	15	0	5	[100, 100]
200/RC2_2_1	100	50	20	0	5	[100, 100]
200/RC2_2_1	150	50	20	0	5	[100, 100]

Table 5.29: Table 4 of all benchmark instances. The instance names show *original instance size/used instance.*